UNIVERSIDAD CENTRAL DEL ECUADOR

FACULTAD DE INGENIERÍA, CIENCIAS FÍSICAS

Y MATEMÁTICA

CARRERA DE INGENIERÍA INFORMÁTICA

SISTEMA DE INVENTARIO PARA EL MANEJO DE IMPLEMENTOS DEPORTIVOS DE LA FACULTAD DE CULTURA FÍSICA DE LA UNIVERSIDAD CENTRAL DEL ECUADOR.

Trabajo de Graduación previo a la obtención del Título de Ingeniero Informático

AUTOR: Sosa Betancourt José Xavier

TUTOR: Ing. César Augusto Morales Mejía, Msc

QUITO – ECUADOR

2014

DEDICATORIA

Este proyecto en especial se lo dedico a Dios por brindarme la sabiduría para ir sorteando cada paso de mi vida y permitirme culminar mi carrera profesional.

A mi madre que con su bendición desde el cielo me brindo la fuerza para seguir luchando para culminar mi objetivo.

A mi padre por ser mi guía y apoyo en todos los buenos y malos momentos de mi vida.

A mi hermana por ser mi amiga, mi fuerza y mi inspiración para culminar mi meta.

Xavier Sosa B.

AGRADECIMIENTO

Al finalizar este trabajo quisiera reconocer en primer lugar a Dios por permitir la realización de este proyecto que es la culminación de una etapa mas de mi vida, a mi padre que supo inculcarme la ética y los valores que estarán presentes en mi vida personal asi como profesional, a mi hermana por estar presente en cada instante de mi vida brindandome su apoyo.

Un especial reconocimiento a mi tutor de tesis, el Magister César Morales que gracias a su guía y apoyo supo encaminarmepara la culminación de este proyecto.

A cada profesor en la Escuela de Ciencias, quienes con su dedicación y entusiasmo me guiaron en todo el transcurso de mi vida universitaria.

Finalmente quisiera agradecer a mi familia, a mi novia, a misamigos y a todas aquellas personas que formaron parte en este proyecto asi como en mi formación personal.

AUTORIZACIÓN DE LA AUTORIA INTELECTUAL

Yo, José Xavier Sosa Betancourt en calidad de autor del trabajo de tesis realizada sobre, Sistema de Inventario para el Manejo de Implementos Deportivos de la Facultad de Cultura Física de la Universidad Central del Ecuador, por la presente autorizo a la UNIVERSIDAD CENTRAL DEL ECUADOR, hacer uso de todos los contenidos que me pertenecen o de parte de los que contienen esta obra, con fines estrictamente académicos o de investigación.

Los derechos que como autor me corresponden, con excepción de la presente autorización, seguirán vigentes a mi favor, de conformidad con lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la ley de la Propiedad Intelectual y su Reglamento.

Quito, a los 19 días del mes de Mayo del 2014.

José Xavier Sosa Betancourt

CI: 1715626360

APROBACIÓN DEL DIRECTOR DE TESIS

FACULTAD DE INGENIERÍA, CIENCIAS FÍSICAS Y MATEMÁTICA
CARRERA DE INGENIERÍA INFORMÁTICA

Quito D.M. 15 de Abril del 2014

Señor Ingeniero
Boris Herrera
DIRECTOR DE CARRERA DE INGENIERÍA INFORMÁTICA
Presente.-

Señor Director:

Yo. Ingeniero César Augusto Morales Mejía, Docente de la Carrera de Ingeniería Informática de la Facultad de Ingeniería Ciencias Físicas y Matemática de la Universidad Central del Ecuador, en calidad de Director de Tesis del proyecto de investigación, análisis y diseño delSistema de Inventario para el Manejo de Implementos Deportivos de la Facultad de Cultura Física de la Universidad Central del Ecuador, presentado y desarrollado por José Xavier Sosa Betancourt, paraaprobar el tema de trabajo de graduación previo a la obtención del título deIngeniero Informático, considero que el proyecto reúne los requisitosnecesarios.

Por la atención que digne al presente, reitero mi agradecimiento.

Atentamente,

Ing. César Augusto Morales Mejía, Mcs
DIRECTOR DE TESIS

CERTIFICACIÓN DEL DIRECTOR DE TESIS

FACULTAD DE INGENIERÍA, CIENCIAS FÍSICAS Y MATEMÁTICA
CARRERA DE INGENIERÍA INFORMÁTICA

Quito D.M. 15 de Abril del 2014

Señor Ingeniero
Boris Herrera
DIRECTOR DE CARRERA DE INGENIERÍA INFORMÁTICA
Presente.-

Señor Director:

Yo. Ingeniero César Augusto Morales Mejía, Docente de la Carrera de Ingeniería Informática de la Facultad de Ingeniería Ciencias Físicas y Matemática de la Universidad Central del Ecuador.

Certifico:

Luego de las revisiones técnicas realizadas por mi persona al desarrollo del proyecto de investigación **"SISTEMA DE INVENTARIO PARA EL MANEJO DE IMPLEMENTOS DEPORTIVOS DE LA FACULTAD DE CULTURA FÍSICA DE LA UNIVERSIDAD CENTRAL DEL ECUADOR"**llevado a cabo por parte del egresado de la Carrera de Ingeniería Informática, Señor **José Xavier Sosa Betancourt**, con CC. 1715626360 ha concluido, consecuentemente el mencionado estudiante podrá continuar con los trámites de graduación correspondiente de acuerdo a lo que estipula las normas y disposiciones legales.

Por la atención que digne al presente, reitero mi agradecimiento.

Atentamente,

Ing. César Augusto Morales Mejía, Mcs
DIRECTOR DE TESIS

CERTIFICADO REVISORES

UNIVERSIDAD CENTRAL DEL ECUADOR
FACULTAD DE INGENIERÍA, CIENCIAS FÍSICAS Y MATEMÁTICA
CARRERA DE INGENIERÍA INFORMÁTICA, DISEÑO INDUSTRIAL, MATEMÁTICA, Y COMPUTACIÓN GRÁFICA.

DIRECCIÓN

Teléfonos: 2542026 - 2236987 ext. 218

Quito, 21 de mayo de 2014.
Oficio 121-2014 DC-IINF.

Ingenieros
Mauro Rosas
Jorge Gordillo
DOCENTES DE LA CARRERA DE
INGENIERÍA INFORMÁTICA
Presente.-

Señores Docentes:
A fin de dar cumplimiento a lo dispuesto en el "Reglamento para la Obtención de los Títulos Profesionales en la Facultad de Ingeniería, Ciencias Físicas y Matemática", aprobado por el H. Consejo Universitario, en sesión del 31 de octubre de 2011; agradeceré a usted, calificar el Trabajo de Graduación titulado: "**SISTEMA DE INVENTARIO PARA EL MANEJO DE IMPLEMENTOS DEPORTIVOS DE LA FACULTAD DE CULTURA FÍSICA DE LA UNIVERSIDAD CENTRAL DEL ECUADOR**" realizado por el (la) estudiante: JOSÉ XAVIER SOSA BETANCOURT requisito previo a la obtención del título de INGENIERO (A) INFORMATICA, en base al Formulario del Resultado del Trabajo de Graduación, que me permito remitirle.

Este formulario, deberá enviarse a la Secretaría General de la Facultad en un plazo no mayor a ocho días.

Atentamente,

Ing. Boris Herrera Flores MSc.
DIRECTOR, CARRERA INGENIERÍA EN INFORMÁTICA.

RECIBI CONFORME	FIRMA
Ing. Mauro Rosas	
Ing. Jorge Gordillo	

Sol. No. C-481

CALIFICACIÓN REVISORES

UNIVERSIDAD CENTRAL DEL ECUADOR
FACULTAD DE INGENIERÍA, CIENCIAS FÍSICAS Y MATEMÁTICA
CARRERA DE INGENIERÍA INFORMÁTICA

DIRECCIÓN

Teléfonos: 2542026 - 2236987 ext. 218 Fax 2226039

RESULTADO DEL TRABAJO DE GRADUACIÓN

CARRERA DE: **INGENIERÍA INFORMÁTICA**

Quito, junio 10 de 2014

Señor: JOSÉ XAVIER SOSA BETANCOURT.

TEMA: "**SISTEMA DE INVENTARIO PARA EL MANEJO DE IMPLEMENTOS DEPORTIVOS DE LA FACULTAD DE CULTURA FÍSICA DE LA UNIVERSIDAD CENTRAL DEL ECUADOR.**"

CALIFICACIÓN:

TRIBUNAL	PROFESOR (A)	NOTA SOBRE VEINTE		FIRMA
		NUMERO	LETRAS	
PROFESOR TITULAR	Ing. Mauro Rosas	19	Diecinueve	
PROFESOR TITULAR	Ing. Jorge Gordillo	18	Dieciocho	
PROMEDIO				

Dra. Ruth Flores Chacón
SECRETARIA ABOGADA

CONTENIDO

LISTADO DE FIGURAS

RESUMEN

SISTEMA DE INVENTARIO PARA EL MANEJO DE IMPLEMENTOS DEPORTIVOS DE LA FACULTAD DE CULTURA FÍSICA DE LA UNIVERSIDAD CENTRAL DEL ECUADOR.

La necesidad de llevar un control de los accesorios deportivos y materiales con los que cuenta la Facultad de Cultura Física de la Universidad Central del Ecuador, requiere la implementación de una aplicación informática que permita administrar los mencionados accesorios y materiales, facilitando además la gestión de personas que están a cargo.

El presente trabajo proporciona un sistema, el cual permite a través de la web, el registro de usuarios, accesorios y materiales de la Facultad y el ingreso de solicitudes de préstamo con una respuesta ágil y segura.

El sistema realiza un control del stock basándose en la cantidad de ingresos, devoluciones y préstamos. Además el sistema esta diseñado para que almacene un historial de los cambios que se hagan en la información, la cual se grabará en una base de datos.

DESCRIPTORES:

ADMINISTRACIÓN DE BODEGA / DESARROLLO ITERATIVO INCREMENTAL / FACULTAD DE CULTURA FÍSICA / JAVA / JBOSS / SQL

ABSTRACT

INVENTORY SYSTEM FOR HANDLING SPORTS IMPLEMENTS OF THE PHYSICAL CULTURE SCHOOL OF THE CENTRAL UNIVERSITY OF ECUADOR.

The need to take a control of sports accessories and materials are there in the Physical Culture school of the Central University of Ecuador, requires a computer application that allows administer the aforementioned accessories and materials, further helping the management of people who are in charge.

The present work provides a system, which allows through the web, the users' record, accessories and materials of the school and income loan requests with a quick and safe response.

The system controls stock based on the amount of income, returns and loans. In addition the system this one designed in order that it stores a record of the changes that are done in the information, which will be recorded in a database.

DESCRIPTORS:

WAREHOUSEMANAGEMENT/ITERATIVEINCREMENTALDEVELOPMENT/FACULTY OFPHYSICAL CULTURE/JAVA /JBOSS/ SQL

CERTIFICADO DE TRADUCCIÓN

Yo, Lic. **PACO JAVIER PAREDES ROSAS**, con C.I. 0501847602, persona certificada en el idioma **INGLÉS** según certificado emitido por el instituto **P.R.A.L.I.** (Public Relations And Language Institute) y avalado por el ministerio de Educación y Cultura,

CERTIFICO:

Que, el texto resumen de la tesis del Sr. **JOSÉ XAVIER SOSA BETANCOURT,** con C.I. 1715626360, a petición suya, fue traducido al idioma inglés por mi persona.

Autorizo al señor antes mencionado, hacer uso de este certificado como a bien necesite.

Quito, a los 18 días del mes de marzo del 2014.

Lic. Paco Javier Paredes Rosas
C.I. 0501847602

MINISTERIO DE EDUCACION

CONFIERE

A **PAREDES ROSAS PACO JAVIER** EL

CERTIFICADO DE CAPACITACION OCUPACIONAL

POR HABER APROBADO EL CURSO DE: **INGLÉS AVANZADO**

EN EL CENTRO OCUPACIONAL: **PRALI. SUR**

DEL 03 de Octubre 11 AL 31 de Enero DE L 2012 POR 160 HORAS.

Lugar y fecha: Quito, Enero 31 del 2012

DIRECCION PROVINCIAL DE EDUCACION DE PICHINCHA
REGIMEN ESCOLAR

Jefe de la División Provincial
Educación Popular Permanente

Jorge Cabezas Gallegos
JEFE DEL DPTO. REGIMEN ESCOLAR Y
REFRENDACION DE TITULOS (E)

PUBLIC RELATIONS AND LANGUAGE INSTITUTE
PRALIS
DIRECCION

Director de
Centro Ocupacional

Lcdo. Oscar Villavicencio

P. R. A. L. I.

Prestigious and Experience at your Service

CERTIFICATE

This is to certificate that

PAREDES ROSAS PACO JAVIER

has attended and completed the

ENGLISH COURSE

In English during 570 hours with the final average of 17

Quito, March 18th, 2011

PUBLIC RELATIONS AND LANGUAGE INSTITUTE
PRALI

General Director

Academic coordinator

CAPITULO I

1. Introducción

En la actualidad surge la necesidad de contar con servicios de tecnología que ayuden a complementar el trabajo manual apoyados en sistemas informáticos los cuales sean de calidad y estén enfocados en las necesidades o requerimientos delas instituciones u organizaciones y cumplan sus expectativas.

Los sistemas informáticos han adquirido importancia significativa en la administración y control de las diferentes operaciones que se llevan a cabo dentro de las organizaciones; cuya aplicación se ve reflejada en la utilización eficiente de recursos materiales, humanos y sobre todo económicos.

De ahí la importancia de aplicar sistemas informáticos para el control y manejo de inventarios ya que ayudará a tener una gestión adecuada de los bienes que posea una institución u organización y con ello se podrá llevar un registro detallado y evitar tener bajas en los implementos lo que afectará en diversos aspectos.

Al tener un sistema de Inventario podemos llevar un adecuado control de todos los implementos que se posea así como un registro de las personas responsables de dichos implementos.

El llevar un control actualizado y constante del stock de implementos con los que cuenta la institución u organización permitirá ampliar la visión de quienes están a cargo de los mismos, facilitando la elaboración de proyecciones y asignación del presupuesto para la adquisición o reemplazo de los mismos.

Al sistematizar el inventario ayudará a realizar informes completos con los cuales se podrá tener una visión clara de cuáles son los implementos existentes en bodega y los que están siendo utilizados para realizar determinadas actividades, lo que facilitará el control de los bienes y de quienes estén a cargo de los mismos.

De esto se puede decir que el sistema de inventario a realizarse está enfocado y es aplicable a cualquier tipo de organización independientemente de su tamaño, sector o tipo de servicio, el resultado debe ser un sistema confiable, eficientemente implementado, con alta calidad y dentro de los costos esperados.

1.1. Reseña Histórica de la Universidad Central del Ecuador.

El Congreso de Cundinamarca da paso a la fundación de las universidades centrales, en las capitales de los departamentos que integraban la Gran Colombia lo que haría posible la posterior institucionalización de la Universidad Central del Ecuador luego que el 13 de marzo de 1830 se proclamara al Ecuador como república independiente.

La Universidad Central del Ecuador remonta sus orígenes al año de 1836 mediante la unión de celebres y reconocidas Universidades de la época el Seminario de San Luis y San Gregorio Magno fundada en 1651 por los Jesuitas, la Santo Tomás de Aquino, fundada en 1681 por los padres Dominicos y la San Fulgencio fundada en 1586 por los Agustinos.

Se fundamenta sobre la base de Real Universidad Publica Santo Tomas con el nombre de Universidad Central de Quito hasta que por decreto del entonces presidente Vicente Rocafuerte se cambia la palabra Quito por Ecuador tomando el nombre definitivo de Universidad Central del Ecuador.

Siendo la más antigua y grande de la República del Ecuador, su sede principal esta ubica en el centro - norte de la ciudad de Quito, en la llamada ciudadela universitaria, además de comprender sus sedes en el Sur de Quito, en la ciudad de Santo Domingo de los Colorados, y en las Islas Galápagos. Afiliada desde 2012 a la Red Ecuatoriana de Universidades para Investigación y Postgrados.

En el seno de las ideas independentistas del siglo XVIII, nace la figura de la Universidad Central de Quito caracterizándose por una larga trayectoria histórica-política en el Ecuador y en el Continente evidenciada por su

huella en el quehacer académico y en la producción de pensamiento que escriben el destino socio-económico y cultural de la Patria.

1.1.1.Historia De La Facultad De Cultura Física

La Facultad de Cultura Física de la Universidad Central del Ecuador ha ido evolucionando paulatinamente hasta convertirse en la actualidad en una de las facultades más jóvenes de la universidad, sin embargo su trayectoria inicia en el año de 1919 cuando se crea la Liga Deportiva Universitaria y con ella comienza el deporte en la Universidad Central.

En el año 1954 el profesor Genaro Fierro sería nombrado profesor de Cultura Física de la universidad por el Honorable Consejo Universitario, después de impulsar la creación de actual estadio universitario.

El 3 de mayo de 1955 la Cultura Física se incluye como materia obligatoria para todas las facultades y por decreto del Dr. Alfredo Pérez Guerrero y el Dr. Cesar Aníbal Espinosa, rector y vicerrector de la universidad respectivamente se crea el Instituto de Cultura Física.

La primera asamblea de profesores confirmo al Instituto el nombre de Alfredo Pérez Guerrero en el año de 1956 se aprueban planes y programas para el entonces nuevo instituto de Cultura Física.

La facultad de filosofía toma la creación de la escuela de Cultura Física siendo esta la escuela con más de 50 años de vida. Se separa de la facultad de filosofía y actualmente cuenta son un elemento humano de 1200 alumnos, 31 profesores que laboran en las actividades de: acondicionamiento físico, atletismo, basquetbol, futbol, judo, levantamiento de pesas, taekwondo, tenis y voleibol.

La facultad de cultura física, como unidad académica superior que forma parte de la Universidad Central del Ecuador, tiene como objetivo formar profesionales en nivel superior en el campo de la Cultura Física, deportes y recreación para satisfacer las necesidades del sistema educativo del país y de la comunidad, mediante una sólida formación científica, tecnológica, humanística y con respeto al medio ambiente.

1.2. Relación Funcional del Departamento de Administración de bodega y los Docentes de la Facultad de Cultura Física de la UCE.

1.2.1. Relación Funcional entre docentes y el personal de bodega

El departamento de Administración de bodega de la Facultad de Cultura Física de la UCE atiende diariamente solicitudes de implementos deportivos por parte de los docentes, ya que son requeridos para el desarrollo de las diferentes actividades que se llevan a cabo con los alumnos en las clases impartidas.

1.2.2. Funciones realizadas por el administrador de bodega.

a) Registrar el ingreso de nuevos implementos deportivos ya sean adquiridos o donados.
b) Elaborar informes de los implementos deportivos que necesitan mantenimiento.
c) Registrar las solicitudes de implementos deportivos realizadas por los docentes.
d) Elaborar un plan anual para la adquisición de nuevos implementos deportivos de acuerdo a las necesidades.
e) Elaborar informes de las personas que se encuentran a cargo de los implementos deportivos.
f) Elaborar informes anuales para las autoridades con el objetivo de dar de baja a los implementos deportivos que estén obsoletos.

1.3. Presentación del Problema

1.3.1. Planteamiento del Problema

Al realizar un análisis del estado actual del manejo de bodegas de implementos deportivos se encontraron varios problemas entre los cuales se plantea los siguientes:

Operaciones Manuales.

Inconsistencia de Información.

Discrepancias en la administración de las bodegas.

Sin registro de responsables.

Pérdida de implementos deportivos.

No confiabilidad, ni consolidación de información.

Retraso en la entrega de informes.

Estos problemas detallados anteriormente se deben a no poseer información detallada, precisa y fiable de todos los elementos que están dentro del sistema de inventario como son: Administrador de bodega, responsables de los implementos deportivos, implementos deportivos sin registro, entre otros, por esta razón existe la necesidad de implementar un SISTEMA DE INVENTARIO PARA EL MANEJO DE IMPLEMENTOS DEPORTIVOS el cual ayudará a corregir y satisfacer las necesidades de los Autoridades, Administradores de bodega, Educadores y Estudiantes que están inmersos en las diferentes actividades tanto académicas como de competencia que organiza la Facultad de Cultura Física de la Universidad Central del Ecuador.

1.3.2. Formulación del Problema

¿Cuál es el resultado de no administrar y gestionar de manera adecuada los implementos deportivos con los que cuenta la Facultad de Cultura Física de la Universidad Central del Ecuador?

La Facultad de Cultura Física de la Universidad Central del Ecuador no cuenta con información detallada, fiable y actualizada de los implementos deportivos que posee, es por esta razón que se desarrollará un sistema para automatizar el Manejo del Inventario que permita llevar un Control de los implementos deportivos que la Facultad de Cultura Física de la Universidad Central del Ecuador posee, llevando un registro de los implementos deportivos así como de las personas responsables de los mismos y esto ayudará a una administración eficiente y eficaz.

1.4. Objetivos de la Investigación

1.4.1. Objetivo General

- Implementación de un Sistema vía web para el manejo de los implementos deportivos que la Facultad de Cultura Física de la Universidad Central del Ecuador posee.

1.4.2. Objetivos Específicos

- Crear un aplicativo web mediante herramientas de licenciamiento libre Proporcionar que permita hacer solicitudes en línea.
- Obtener la disponibilidad de los materiales y/o implementos deportivos en tipo real mediante el aplicativo web.
- Facilitar la carga masiva de materiales y/o implementos deportivos mediante el aplicativo web.

1.5. Justificación del Tema

1.5.1. Impacto Social

El desarrollo de la tesis servirá como modelo de implementación de un sistema informático que permita a llevar un control detallado de los bines activos para cualquier institución u organización que cuente con un almacén o bodega tanto de implementos deportivos u objetos similares.

1.5.2. Impacto Económico

El desarrollo de la tesis será en herramientas OPEN SOURCE, es decir no tendrá costos de licenciamiento.

Los equipos utilizados para la implementación del sistema serán facilitados por las autoridades de la Facultad de Cultura Física de la UCE, debido a que estos equipos permanecerán en el área de bodega para el uso constante del sistema implementado.

Los recursos materiales y tecnológicos que se utilicen para el desarrollo del sistema serán por autogestión.

1.5.3. Impacto Técnico.

Al realizar la aplicación para el manejo vía Web hace que su acceso sea posible por medio de la red que la Universidad Central del Ecuador posee, tomando en cuenta que se debe realizar una inversión en un servidor para la instalación de la aplicación considerando que el desarrollador está capacitado para realizar el mencionado proyecto.

1.6. Limitaciones del proyecto

Dentro de las limitaciones podemos enfatizar los siguientes aspectos.

- El aplicativo web no está diseñado para validar el tipo de codificación asignada a los materiales o implementos deportivos.
- Para que un usuario pueda utilizar el aplicativo web deberá ser registrado por un administrador.
- El formato de archivo para la carga masiva de materiales y/o implementos deportivos debe tener una extensión xls.
- El aplicativo no está diseñado para realizar respaldos automáticos de la base de datos, por lo que se lo tendrá que hacer manualmente.

1.7. Alcance del proyecto

La propuesta de desarrollo es la implementación de un sistema de inventario dirigida para la Facultad de Cultura Física de la Universidad Central del Ecuador, este sistema está enfocado en el manejo de los implementos deportivos que la mencionada facultad posee, llevando un control detallado del stock de implementos deportivos, las personas que se encuentran a cargo, periodos de obsolescencia.

La información requerida será entregada por medio de informes los cuales serán el resultado de determinada búsqueda de acuerdo a la necesidad requerida.

Esta propuesta está dirigida a todas las organizaciones y/o instituciones que requieran un sistema de inventario para el manejo de implementos deportivos o similares.

El sistema estará desarrollado con un enfoque genérico que facilite la adaptación del mismo en cualquier institución de similares características que así lo requiera.

1.8 Flujo de Trabajo

En la Figura 1., se presenta una puesta del proceso a realizarse cuando el sistema se encuentre en marcha.

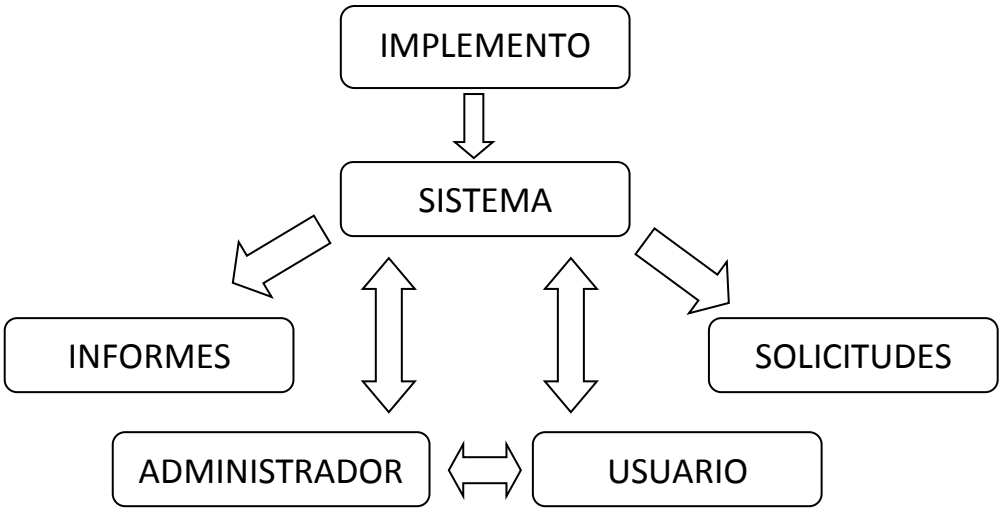


Figura. 1. Esquema Flujo de Trabajo
Autor : Tesista
Fuente : Tesista

a) Los implementos deportivos sean adquiridos, donados o que se encuentren en stock deberán ser ingresados al sistema.
b) El sistema solicitara requerimientos para el ingreso de los implementos, tales como:
    a. Claves de usuario.
    b. Fecha de adquisición.
    c. Código.
    d. Descripción del implemento.
    e. Tiempo estimado de obsolescencia.
    f. Responsable.
    g. Observaciones.
c) El administrador del sistema estará facultado para realizar:
    a. Ingreso de usuarios.
    b. Ingreso de implementos deportivos.
    c. Elaboración de informes.
    d. Aprobación de solicitudes.
    e. Actas de recepción de implementos deportivos.

d) El usuario del sistema estará facultado para realizar:
    a. Solicitudes de implementos deportivos.
    b. Consulta de implementos disponibles.
    c. Acta de entrega de implementos deportivos.
e) El sistema presentará los siguientes informes:
    a. Stock actual en bodega.
    b. Obsolescencia de los implementos deportivos.
    c. Implementos deportivos que se están utilizando y sus responsables.
f) Las solicitudes de implementos deportivos deberán ser elaboradas por el usuario del sistema, aprobadas por el administrador e impresas.

CAPITULO II

2. Técnicas y Herramientas:

2.1. Metodología de Desarrollo:

La metodología que se va a utilizar para desarrollar la aplicación será el método Iterativo e Incremental. A continuación se detallarán aspectos importantes de este método.

2.1.1. Desarrollo iterativo e incremental.

Al tomar como método el desarrollo iterativo e incremental al proyecto se lo divide en varios bloques temporales llamados iteraciones.

A cada iteración se lo puede ver como un mini proyecto, el proceso de trabajo realizado en cada mini proyecto se lo repite de forma similar para cada uno de ellos y por ello al finalizar se pueda entregar un trabajo completo y sólido. Por la serie de repeticiones toma el nombre de "iterativo".

Al tener subdividido el proyecto el cliente obtiene el beneficio de ir revisando continuamente el desarrollo del proyecto con lo que se puede ir avanzando ("forma incremental") y de esta manera no se deja para el final del proyecto ninguna actividad arriesgada relacionada con la entrega de requisitos.

Al ir realizando cada iteración (mini proyecto) el desarrollador va avanzando en el proyecto basándose en los resultados completados en las iteraciones anteriores y añadiendo nuevos objetivos/requisitos o mejorando los que ya fueron completados.

También en esta metodología se puede decir que se fundamenta en la priorización de los objetivos/requisitos en función del valor que aportan al cliente.

En todo el transcurso del desarrollo del proyecto se tiene una serie de fases que al unirse se obtiene como resultado final el sistema requerido,

cada fase está compuesta por un número de iteraciones las cuales generan versiones del sistema.

Las fases son 4:

- Planificación: Define el ámbito y objetivos del proyecto, la funcionalidad y capacidades del producto.
- Construcción: El producto se desarrolla a través de iteraciones donde cada iteración involucra tareas de análisis, diseño e implementación. Las fases de estudio y análisis sólo dieron una arquitectura básica que es aquí refinada de manera incremental conforme se construye (se permiten cambios en la estructura), gran parte del trabajo es programación y pruebas, se documenta tanto el sistema construido como el manejo del mismo. Esta fase proporciona un producto construido junto con la documentación.

- Ejecución: Tanto la funcionalidad como el dominio del problema se estudian en profundidad, se define una arquitectura básica y se planifica el proyecto considerando recursos disponibles.
- Actuar: Se finaliza el proyecto y se entrega al usuario para un uso real, los manuales de usuario se completan y refinan con la información anterior.

Figura. 2. Ciclos iterativos de desarrollo

Autor : Factor libre
Fuente : http://factorlibre.com/precio

2.1.1.1. Características:

Usando análisis y mediciones como guías para el proceso de mejora es una diferencia mayor entre las mejoras iterativas y el desarrollo rápido de aplicaciones, principalmente por dos razones:

- Provee de soporte para determinar la efectividad de los procesos y de la calidad del producto.
- Permite estudiar y después mejorar y ajustar el proceso para el ambiente en particular.

Estas mediciones y actividades de análisis pueden ser añadidas a los métodos de desarrollo rápido existentes.

De hecho, el contexto de iteraciones múltiples conlleva ventajas en el uso de mediciones. Las medidas a veces son difíciles de comprender en lo absoluto, aunque en los cambios relativos en las medidas a través de la evolución del sistema puede ser muy informativo porque proveen una base de comparación.

Por ejemplo, un vector de medidas m1, m2,..,mn puede ser definido para caracterizar varios aspectos del producto en cierto punto, como puede ser el esfuerzo total realizado, los cambios, los defectos, los atributos lógicos, físico y dinámico, consideraciones del entorno, etcétera. Así el observador puede decir como las características del producto como el tamaño, la complejidad, el acoplamiento y la cohesión incrementan o disminuyen en el tiempo. También puede monitorearse el cambio relativo de varios aspectos de un producto o pueden proveer los límites de las medidas para apuntar a problemas potenciales y anomalías.[1]

2.1.1.2. Ventajas del desarrollo iterativo e incremental.

- Reduce riesgos del proyecto basándose en una retroalimentación temprana.
- Incorpora objetivos de calidad.

[1] http://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente

- Mayor flexibilidad para manejar cambios nuevos o modificaciones a los mismos.
- Integra el desarrollo con el mantenimiento.
- La complejidad nunca resulta abrumadora.
- Se produce retroalimentación en una etapa temprana, porque la implementación se efectúa rápidamente con una parte pequeña del sistema.[2]

2.2 Sistema de Inventario

Antes de iniciar en el desarrollo de un sistema de inventarios, es conveniente definir que es un inventario.

2.2.1 Concepto de inventario

Es el Documento que contiene la relación pormenorizada de los bienes muebles e inmuebles que posee una entidad, en el cual debe estar detallado el nombre y código patrimonial, características propias, estado actual de conservación, valor en libros, valor de tasación, usuario, ubicación y uso del bien.[3]

Es el control de las existencias de materiales, equipos, muebles e inmuebles con que cuenta una dependencia o una entidad, es un documento o sistema donde se lleva un control para el manejo administrativo de los materiales.

2.2.2 Tipos de Inventario.

Dentro de la clasificación del inventario se consideran tres tipos:

- Materia prima.
- Producto en proceso.
- Producto terminado.

[2] http://es.scribd.com/doc/62914255/30/Figura-10-Desarrollo-iterativo-e-incremental (Ingeniería del Software: Metodologías y Ciclos de Vida, Laboratorio Nacional de Calidad del Software de INTECO (España).)
http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rea_c_ji/capitulo3.pdf
[3]http://www.americanadeavaluos.com/index.php?option=com_content&view=article&id=5&Itemid=6

a. **Materia Prima.-** Son los productos que van a ser usados para la línea de producción. Estos deben ser modificados y transformados para convertirse en producto final.
b. **Producto en Proceso.-** Son prácticamente convertidos en producto final pero están en producción en proceso.
c. **Producto Terminado.-** Es el producto final en espera de ser vendido o distribuido.

2.2.3. Control

Dentro de la elaboración de este proyecto uno de los aspectos importantes es el de controlar y a continuación se definirá este término como:

Es un mecanismo preventivo y correctivo adoptado por la administración de una dependencia o entidad que permite la oportuna detección y corrección de desviaciones, ineficiencias o incongruencias en el curso de la formulación, instrumentación, ejecución y evaluación de las acciones, con el propósito de procurar el cumplimiento de la normatividad que las rige, y las estrategias, políticas, objetivos, metas y asignación de recursos.[4]

2.2.4. Propiedades del inventario

Dentro de un inventario existen varias propiedades de entre las cuales podemos nombrar las siguientes:

- Demanda.
- Productos.
- Horizonte de planeación.
- Costos.

a. **Demanda.-** Son las unidades requeridas que se toman del inventario en un determinado tiempo.

[4] http://www.definicion.org/control

b. **Productos.-** Los productos pueden ser uno o varios. Así mismo se pueden clasificar por unidad o por lote dependiendo del proceso; perecederos o duraderos dependiendo de su vida útil, divisibles o indivisibles, etc.
c. **Horizonte de planeación.-** Es el periodo de tiempo durante el cual el nivel de inventario debe ser controlado.
d. **Costos.-**Es el gasto económico que representa la fabricación de un producto o la prestación de un servicio.

2.2.5. Objetivo de un inventario.

El objetivo principal del control del inventario es tener la cantidad apropiada de materiales u otros productos en el lugar adecuado, el tiempo oportuno y con el menor costo posible.

Dentro de los factores de costo dentro del control del inventario se pude señalar los siguientes:

2.2.5.1. Costo de adquisición

El costo de adquisición de un bien de uso representa el sacrificio económico para adquirir el bien y ponerlo en condiciones de ser utilizado en la actividad.

Las normas contables profesionales vigentes disponen que para determinar el costo de adquisición se debe considerar el precio que debe pagarse al contado. Por lo tanto, toda diferencia con dicho precio será reconocida como componente financiero implícito y el descuento por pronto pago que se obtuviera significará una reducción del cargo financiero.[5]

2.2.5.2. Costos de no tener inventario

El costo de tener o mantener el inventario en almacenes (H) comprende diferentes conceptos como los de almacenaje, depreciación de bodegas y equipo o renta de estos, impuestos, seguros, desperdicio, obsolescencia, manejo, etc.

[5]http://www.economicas-online.com/bienesde2.htm

2.2.5.3. Costos de no tener inventario de oportunidad

Estos costos pueden tener su origen en faltantes externos cuando a un cliente no se le puede surtir una orden ocasionando órdenes pendientes, disminución en las ventas y pérdida de prestigio comercial, o internos cuando un departamento dentro de la organización no cuenta con materiales o artículos ocasionando pérdidas de producción, retraso en las fechas de entrega.[6]

2.3 Método de Investigación Descriptiva

Para la presente investigación se utilizó el tipo de investigación descriptiva ya que esta tiene como finalidad definir, clasificar, catalogar el objeto de estudio. Este tipo de investigación es aplicable ya que el inventario se lo maneja de forma manual y no se tenía un proceso definido para realizar las prestaciones de materiales y de implementos definidos.

Dentro de la investigación descriptiva nos enfocaremos en el tipo observacional ya que este consiste en registrar el comportamiento del entorno habitual del objeto de investigación.

A continuación se mencionará las características de este tipo de investigación lo cual fue de gran importancia para el levantamiento de requerimientos así como en la definición del proceso y estas son.

- Definición precisa de las condiciones de observación.

- Sistematización y objetividad

- Rigor en el registro del comportamiento.

El método observacional puede ser con o sin intervención lo que ayuda a entender claramente el proceso y su vez intervenir para definir el proceso más adecuado.

2.4. Lenguaje de programación: Java y Tecnología J2EE

2.4.1. Java

“Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de

[6] http://investigaoperativa1.blogspot.com/p/modelo-de-inventarios.html

objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre diciembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es)".[7]

Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más genéricas (objetos) que permitan la reutilización del software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software. Un objeto genérico "cliente", por ejemplo, debería en teoría tener el mismo conjunto de comportamiento en diferentes proyectos, sobre todo cuando estos coinciden en cierta medida.

En este sentido, los objetos podrían verse como piezas reutilizables que pueden emplearse en múltiples proyectos distintos, posibilitando así a la

[7] http://www.ozarate.net/clases/3a_des/PRESENTACION_JAVA.pdf

industria del software a construir proyectos de envergadura empleando componentes ya existentes y de comprobada calidad; conduciendo esto finalmente a una reducción drástica del tiempo de desarrollo.

La reutilización del software ha experimentado resultados dispares, encontrando dos dificultades principales: el diseño de objetos realmente genéricos es pobremente comprendido, y falta una metodología para la amplia comunicación de oportunidades de reutilización. Algunas comunidades de “código abierto” (open source) quieren ayudar en este problema dando medios a los desarrolladores para diseminar la información sobre el uso y versatilidad de objetos reutilizables y bibliotecas de objetos.

La independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, ’’write once, run everywhere’’.

“En la parte del servidor, Java es más popular que nunca, desde la aparición de la especificación de Servlets y JSP (Java Server Pages).

Los servlets y las JSPs supusieron un importante avance ya que:

- El API de programación es muy sencilla, flexible y extensible.
- Los servlets no son procesos independientes (como los CGIs) y por tanto se ejecutan dentro del mismo proceso que la JVM mejorando notablemente el rendimiento y reduciendo la carga computacional y de memoria requeridas.
- Las JSPs son páginas que se compilan dinámicamente (o se pre-compilan previamente a su distribución) de modo que el código que se consigue una ventaja en rendimiento substancial frente a muchos lenguajes interpretados.

La especificación de Servlets y JSPs define un API de programación y los requisitos para un contenedor (servidor) dentro del cual se puedan desplegar estos componentes para formar aplicaciones web dinámicas

completas. Hoy día existen multitud de contenedores (libres y comerciales) compatibles con estas especificaciones. A partir de su expansión entre la comunidad de desarrolladores, estas tecnologías han dado paso a modelos de desarrollo mucho más elaborados con frameworks (pe Struts, Webwork) que se sobreponen sobre los servlets y las JSPs para conseguir un entorno de trabajo mucho más poderoso y segmentado en el que la especialización de roles sea posible (desarrolladores, diseñadores gráficos,...) y se facilite la reutilización y robustez de código. A pesar de todo ello, las tecnologías que subyacen (Servlets y JSPs) son substancialmente las mismas.

Este modelo de trabajo se ha convertido en uno de los estándar de-facto para el desarrollo de aplicaciones web dinámicas de servidor"[8].

2.4.2. Tecnología J2EE

"Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java con arquitectura de N capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación.

Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como u n estándar debido a que los proveedores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por The Java Community Process / JCP.

Java EE incluye varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets

[8]http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

(siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios web. Ello permite al desarrollador crear una Aplicación de Empresa portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel"[9].

2.4.2.1. Arquitectura J2EE

La especificación de J2EE define su arquitectura basándose en los conceptosde *capas*, *containers*, *componentes*, *servicios* ylascaracterística s de cada uno de éstos. Las aplicaciones J2EE son divididas en cuatro capas: la capa cliente, la capa web, la capa negocio y la capa datos. La Figura 3 representa estas capas y las componentes relacionadas.

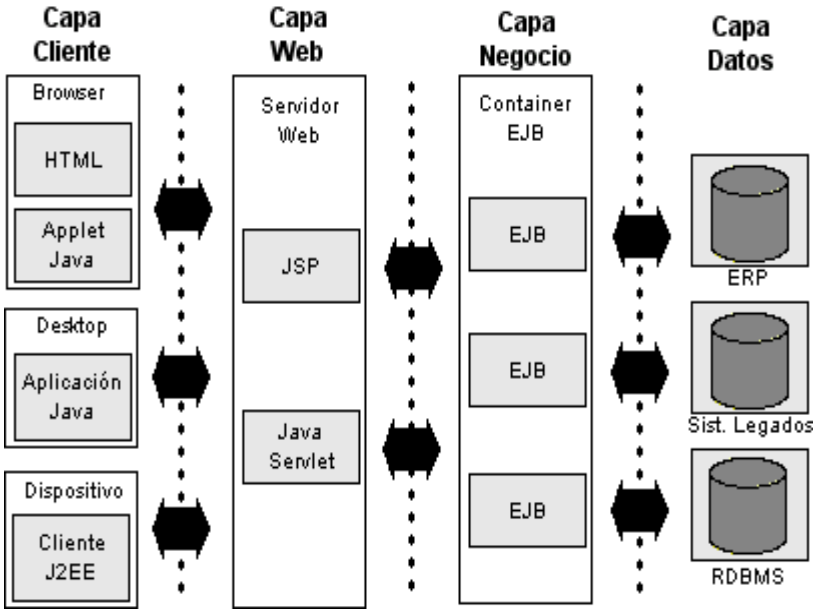


Figura. 3 Arquitectura J2EE

Autor: Juan Manual Barrios N.

Fuente: http://users.dcc.uchile.cl/~jbarrios/J2EE/node14.html

[9]https://es.wikipedia.org/wiki/Java_EE

Capa Cliente

Esta capa corresponde a la interfaz gráfica del sistema y se encarga de interactuar con el usuario. J2EE tiene soporte para diferentes tipos de clientes incluyendo clientes HTML, applets Java y aplicaciones Java.

Capa Web

Esta capa se encuentra en el servidor web y contiene la lógica de presentación que se utiliza para generar una respuesta al cliente. Recibe los datos del usuario desde la capa cliente y basado en éstos genera una respuesta apropiada a la solicitud. J2EE utiliza en esta capa las componentes *Java Servlets* y *JavaServer Pages* para crear los datos que se enviarán al cliente.

Capa Negocio

Esta capa se encuentra en el servidor de aplicaciones y contiene el núcleo de la lógica del negocio de la aplicación. Provee las interfaces necesarias para utilizar el servicio de componentes del negocio. Las componentes del negocio interactúan con la capa de datos y son típicamente implementadas como componentes EJB.

Capa Datos

Esta capa es responsable del sistema de información de la empresa o *Enterprise Information System (EIS)* que incluye bases de datos, sistema de procesamiento datos, sistemas legados y sistemas de planificación de recursos. Esta capa es el punto donde las aplicaciones J2EE se integran con otros sistemas no J2EE o con sistemas legados.

2.4.2.2. Características de la arquitectura J2EE

a. Arquitectura Multicapas

Un modelo multicapa, se define de tal manera que los componentes que forman el sistema se organizarán en **distintas capas, cada una de las cuales tendrá una serie de responsabilidades.** La comunicación entre componentes de las distintas capas se debería realizar a través

de **interfaces**, los cuales deben ser estables y estar bien definidos, de forma que se **facilite así la reutilización y el mantenimiento de los componentes**. Esto conlleva además que cualquier modificación en un componente determinado no afecte a los componentes del resto de capas.

b. Arquitectura basada en componentes

El desarrollo de las aplicaciones se basa en elaborar distintos componentes, los cuales deberán tener un bajo nivel de acoplamiento, facilitando de esta forma la reutilización y portabilidad de los mismos.

c. Arquitectura distribuida

La arquitectura basada en un modelo multicapa conlleva que los componentes de cada una de las distintas capas se pueden empaquetar y ubicar físicamente en distintas máquinas, permitiendo así la distribución del sistema

d. Multiplataforma

Está basada en la utilización del lenguaje Java, el cual es independiente de la plataforma, de forma que las aplicaciones construidas con el mismo se podrán ejecutar en cualquier máquina o dispositivo que permita la ejecución de la máquina virtual de Java.

e. Servicios ofrecidos por contenedores:

Los componentes se ubicarán en distintos contenedores, los cuales se encargarán de gestionarlos. A su vez, los contenedores ofrecen una serie de servicios adicionales, facilitando la construcción y gestión de los distintos componentes

f. Organismo de control

La plataforma Java EE se considera definida por una especificación estándar, realizada por la **JCP – Java Community Process.**

2.5. Tecnologías para el desarrollo de aplicaciones Web

2.5.1 Java Server Faces (JSF)

"Es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones JavaEE. JSF usa Java Server Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL (acrónimo de XML-based User-interface Language, lenguaje basado en XML para la interfaz de usuario)".[10]

2.5.1.1 EL JSF incluye:

a. Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
b. Un conjunto por defecto de componentes para la interfaz de usuario.
c. Dos bibliotecas de etiquetas personalizadas para Java Server Pages que permiten expresar una interfaz Java Server Faces dentro de una página JSP.
d. Un modelo de eventos en el lado del servidor.
e. Administración de estados.
f. Beans administrados.

2.5.1 Arquitectura de un JSP

Su arquitectura define claramente una separación entre las capas de lógica y presentación permitiendo que la conexión entre ambas sea sencilla. Lo que permite a los desarrolladores Web sin experiencia pueden utilizar componentes UI de JSF en sus páginas Web sin escribir código.

JSF proporciona una API basada en componentes que se pueden usar para ensamblar aplicaciones web.

[10] http://es.wikipedia.org/wiki/JavaServer_Faces

Los componentes UI estándar proporcionados por la especificación, están acompañados de “tag libraries” de tipo “core” y “html” (con funcionamiento muy similar a JSTL).

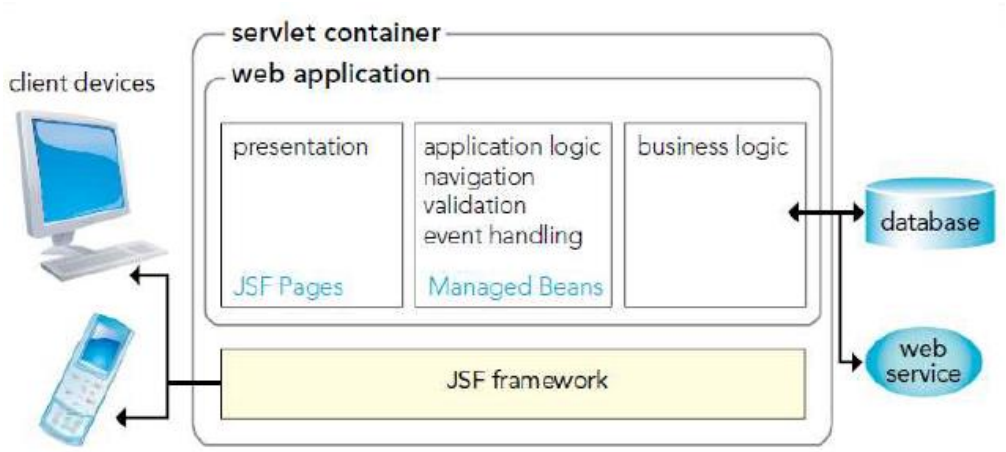


Figura. 4 Arquitectura JSF

Autor: Danner

Fuente: http://icefacesjsfdanner.blogspot.com/

2.5. Servidor de Aplicaciones JBoss

“JBoss es un servidor de aplicaciones J2EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible Java. Los principales desarrolladores trabajan para una empresa de servicios, JBoss Inc., adquirida por Red Hat en abril del 2006, fundada por Marc Fleury, el creador de la primera versión de JBoss. El proyecto está apoyado por una red mundial de colaboradores. Los ingresos de la empresa están basados en un modelo de negocio de servicios.

JBoss implementa todo el paquete de servicios de J2EE”[11].

2.5.1. Características de JBoss

Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.

[11] http://es.wikipedia.org/wiki/JBoss

- Confiable a nivel de empresa
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente
- Servicios del middleware para cualquier objeto de Java
- Ayuda profesional 24x7 de la fuente
- Soporte completo para JMX

2.5.2. Componentes de JBoss

- **EJB 3.0**

Implementa la especificación inicial de EJB 3.0.

- **JBoss AOP**

JBoss AOP está orientado a trabajar con Programación Orientada a Aspectos. Esto permitirá añadir fácilmente servicios empresariales (transacciones, seguridad, persistencia) a clases Java simples.

- **Hibernate**

Hibernate es un servicio de persistencia objeto/relaciones y consultas para Java. Hibernate facilita a los desarrolladores crear las clases de persistencia utilizando el lenguaje Java - incluyendo la asociación, herencia, polimorfismo y composición y el entorno de colecciones Java.

- **JBoss Cache**

JBoss Cache es un producto diseñado para almacenar en caché los objetos Java más frecuentemente accedidos de manera que aumente de forma notable el rendimiento de aplicaciones e -bussines. Eliminando accesos innecesarios a la base de datos, JBoss Cache reduce el tráfico de red e incrementa la escalabilidad de las aplicaciones.

JBoss Cache proporciona dos APIs de caché que se ajustan a nuestras necesidades. La API de JBossCache ofrece una caché tradicional basada en nodos y estructurada en árbol, y la API JBossCacheAOP, edificada

sobre la API de JBossCache, proporciona capacidad para la replicación de objetos Java de grano f ino, con el máximo beneficio del rendimiento.

- **JBoss IDE**

Brinda una IDE Eclipse para el JBoss AS. De esta forma la depuración y otras tareas asociadas al desarrollo de aplicaciones pueden ser realizadas desde el entorno de Eclipse.

- **JBoss JBPM**

Gestor de procesos de negocio, también denominado "WorkFlow".

JBPM es una plataforma para lenguajes de procesos ejecutables, cubriendo desde gestión de procesos de negocio (BPM) bajo workflow hasta orquestación de servicios. Actualmente JBPM soporta tres lenguajes de procesos, cada uno enfocado a un ambiente y funcionalidad especifica:

- jPDL
- BPEL
- Pageflow

jBPM soporta a estos lenguajes de procesos sobre una sola tecnología:

Máquina Virtual de Procesos (PVM).

- **JBoss Portal**

Es una plataforma de código abierto para albergar y servir una interfaz de portales Web, publicando y gestionando el contenido así como adaptando el aspecto de la presentación.

2.5.3 Estructura

bin

Este directorio contiene los ejecutables utilizados por JBoss, el más importante siendo el "script" de arranque utilizado por éste (run.sh).

client

Contiene los archivos JAR's los cuales serán utilizados por los distintos clientes de los EJB's utilizados en el JBoss. Dichos archivos deben ser agregados a la variable CLASSPATH del sistema donde radica el cliente; el cliente generalmente siendo un JSP/Servlet que accesa el EJB, este paradigma gira alrededor de Stubs/Skeletons de RMI una parte central de EJB's.

docs

Este directorio contiene documentación acerca de JBoss.

lib

Este directorio contiene los archivos JAR's empleados por JBoss requeridos en cualquier modalidad.

server

Este directorio contiene tres sub-directorios nombrados: all, default y minimal; cada sub-directorio contiene los distintos archivos de configuración necesarios para ejecutar JBoss en diferentes modalidades.

La modalidad all incluye la ejecución de JBoss para emplearse como "Cluster", ejecución de "Web-Services" y otras funcionalidades más. El directorio default incluye la configuración para ejecutar JBoss de manera básica, mientras el directorio minimal contiene los valores de configuración necesarios para ejecutar JBoss con requerimientos mínimos; el "Script" de arranque proporcionado con JBoss emplea los valores del directorio default, para emplear otra modalidad es necesario modificar dicho "Script" de arranque (run.sh).

A continuación se describen los directorios residentes en la modalidad de arranque default:

conf

Este directorio contiene las diferentes secciones de configuración utilizadas por JBoss, dependiendo de la modalidad utilizada este

directorio puede contener distintos archivos , sin embargo, sus detalles serán descritos en configuración de JBoss .

data

Contiene distintos parámetros y archivos de configuración para las Bases de Datos proporcionadas con JBoss (Hypersonic y la implementación "Messaging" de JBoss) -- generalmente utilizada para aplicaciones demo.

deploy

Este directorio es ampliamente utilizado ya que aquí se colocan los EJB's para que sean ejecutados por JBoss, una vez colocado el archivo JAR (en forma de EJB) en este directorio, JBoss automáticamente expande y ejecuta el EJB.

lib

Contiene los archivos JAR's empleados por JBoss en base a la modalidad tratada.

log

Contiene los distintos registros ("Logs") generados por JBoss.

tmp

Contiene archivos creados por JBoss y utilizados de manera temporal.

work

Contiene las clases y archivos utilizados por JBoss para ejecución.

2.5.4 Directorio Principal de JBoss

Hay un número de subdirectorios bajo JBOSS_HOME, pero los dos más importantes para el desarrollador J2EE típico son el /bin y /server. El directorio /bin contiene el inicio y apagado de JBoss, el /server contiene los directorios en los que eventualmente desplegaremos nuestras aplicaciones.

CAPITULO III

3.1 Análisis de requerimientos.

3.1.1. Levantamiento de requerimientos

3.1.1.1 Análisis previo

La Facultad de Cultura Física de la Universidad Central del Ecuador desde sus inicios llevó su administración de bodegas en forma manual, mediante una orden (verbal o escrita) de ingreso y egreso de implementos, los docentes y estudiantes solicitan su producto, se elabora la nota de ingreso o egreso de bodega.

Revisando las bodegas de forma visual podemos ver que los registros no proporcionan información que es confiable y no llenaba las expectativas de los administradores.

Cuando necesitan revisar el stock, la administrador de bodega solicita al encargado de bodega, que revisen los implementos faltantes para lo cual tienen que ir a bodega y contar, de uno en uno los productos para realizar un informe sobre los implementos perdidos, dañados o que necesitan mantenimiento.

3.1.1.2. Descripción del problema

Entre los principales problemas encontrados se determinaron los siguientes:

Operaciones Manuales.

Inconsistencia de Información.

Discrepancias en la administración de las bodegas.

Imposibilidad de crecimiento.

No confiabilidad de información.

No existe consolidación de información.

3.1.1.3. Función crítica

Los registros que se llevan no constituyen un soporte a la logística operativa en la toma de decisiones, respecto a los productos existentes, afectando la imagen de la Facultad.

3.1.1.4. Tipos de requerimientos

Los siguientes puntos son los requerimientos puntuales realizados por las personas que participan directamente en los procesos de las bodegas de la Facultad de Cultura Física de la Universidad Central del Ecuador, lo cual ayudó a que el sistema pueda a través de sus procesos tener las opciones de reportes, consultas y proceso que realmente necesitan para poder realizar su trabajo de una mejor manera, a continuación se enuncian los principales:

- Control de los ingresos de inventarios.
- Control de los egreso de inventarios.
- Obtener stock actualizados a la fecha.
- Presentación de Información administrativa y de bodega.
- Administración de productos, bodegas, usuarios.

3.1.1.5. Alcance del proyecto

En el alcance del proyecto se enuncian los siguientes:

- Se encuentran todos los asuntos generales de los beneficios que prestara la herramienta.
- Informes que ayudaran al estudio para la adquisición de los productos hasta su colocación efectiva en las bodegas.
- Cubre el manejo del inventario de los implementos ya sean existentes en bodega o a su vez se estén utilizando.
- Disponibilidad de saldos al día.
- Información actualizada de los diferentes movimientos que maneja la bodega.
- Confiabilidad de datos.
- Disponibilidad de información para toma de decisiones.

3.1.1.6. Beneficios

Como beneficios se puede enunciar los siguientes:

- Consolidar los datos de la bodega en una sola base de datos.
- Información de inventarios de bodega.

- Optimizar el tiempo del bodeguero eliminando las transacciones manuales.
- Disminuyendo el tiempo, papel, personal.
- Información de stock minino.
- Información mediante fechas y filtros.

3.1.2. Análisis y diseño del sistema

Dentro del análisis y el diseño se especifican a continuación los detalles que se verán en el sistema.

- Pantallas funcionales: Se detallan las respectivas funciones que cumple cada pantalla funcional en el sistema.
- Procesos generales: Se encuentran los procesos de la bodega y su respectiva narrativa detallando cada paso.
- Diseño de la base de datos: Se explica cómo está estructurada la base de datos con sus modelos Entidad Relación, Relaciones entre tablas, Modelo lógico y físico, Diccionario de datos.
- Proceso de pruebas e implementación

3.2 Descripción del Proceso.

3.2.1 Descripción del Software.

El software permitirá el ingreso de la identificación del docente que realice la solicitud de préstamo, ya sea con su código o documento de identidad.

Posibilitara el registro del tipo de implemento que le fue prestada.

Incluirá la fecha y hora de entrega del implemento deportivo que le fue suministrada y la fecha y hora de entrega.

3.2.2 Diagramación del Sistema

Para la conceptualización de nuestro Sistema nos basamos en las entrevistas con los actores que interviene en el proceso de “Préstamo de implementos deportivos y materiales”.

Al ir directamente a la fuente aseguramos que los requerimientos que fueron levantados cumplen con la funcionalidad necesaria y oportuna que requieren los usuarios para cumplir con sus labores diarias.

3.2.3 Identificación de Actores.

Las personas que interviene en el sistema son:

a) Profesores
b) Administrador de Bodega
c) Administrador de Sistema

3.3. Actores que intervienen en el Sistema

Profesores

- Se trata de un grupo de personas que imparten cátedra en las diferentes asignaturas a ellos encargadas.
- Dichas personas requieren de materiales e implementos deportivos para dictar su clase.
- Son las personas quienes crean una solicitud de préstamo de materiales e implementos.

Administrador de Bodega

- Es la persona encargada de la administración de los materiales e implementos deportivos con los que cuenta la Facultada de Cultura Física de la Universidad Central del Ecuador.
- Es la persona que recepta y atiende solicitudes de préstamo de materiales e implementos deportivos.

Administrador de sistema

- Es la persona encargada de asignar permisos y roles, creación, modificación y eliminación de Usuarios del Sistema, importación y actualización de los materiales e implementos deportivos.

3.4. Requerimientos Funcionales.

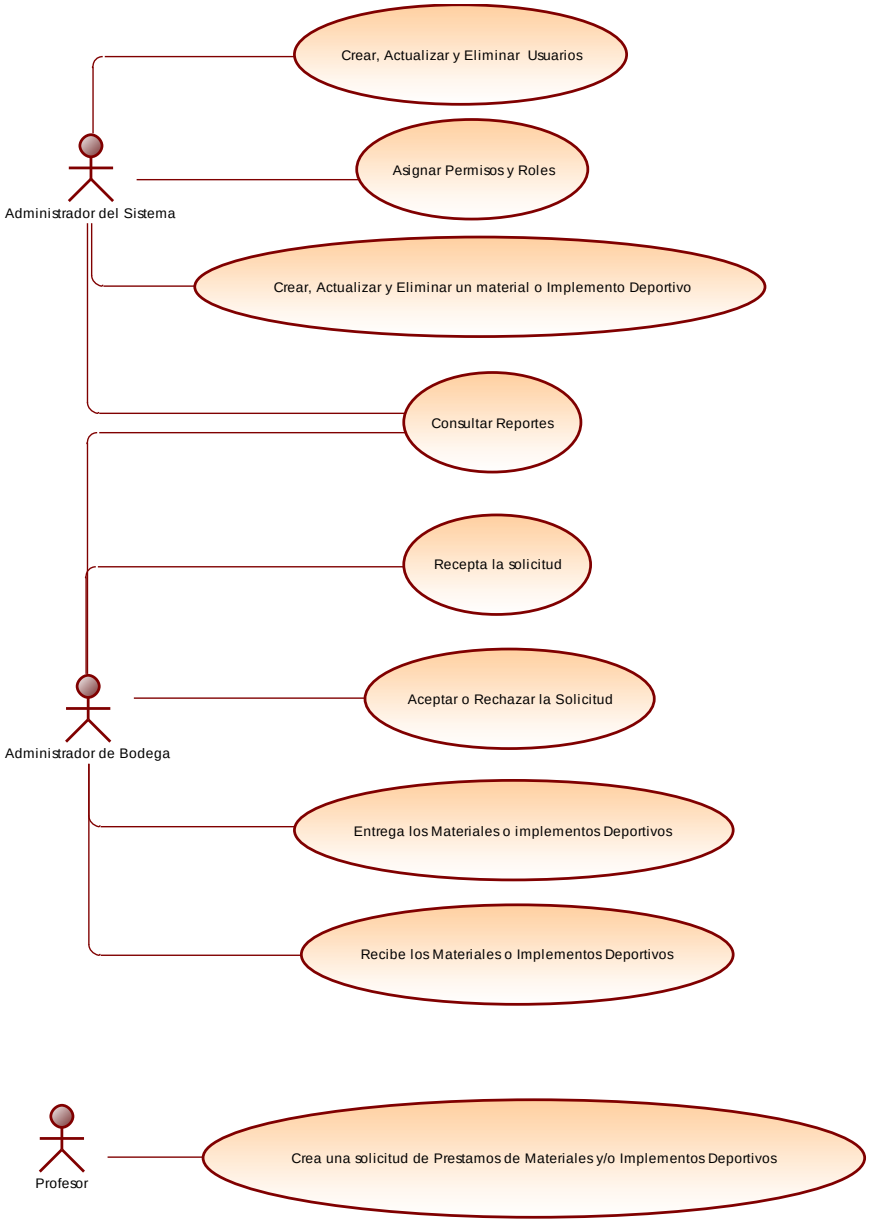
- El sistema permitirá gestionar los implementos que estén disponibles para préstamos.
- El sistema permitirá el ingresar, consultar, eliminar y actualizar un usuario.

- El sistema permitirá que el usuario reserve los implementos deportivos con los que cuenta la Facultad a través de cualquier computador que tenga conexión a la red interna de la Facultad o directamente en el área de bodega de la Facultad de Cultura Física de la Universidad Central del Ecuador.
- El sistema generará reportes de los préstamos de los implementos deportivos así como la existencia de estos cuando sea requerido.

3.5. Requerimientos no Funcionales.

- El sistema correrá Bajo Intranet con el navegador Explorer.
- El sistema estará disponible media hora después de haber sido solicitado el préstamo, si no se va a buscar el implemento deportivo reservado.
- El sistema deberá funcionar en los equipos que actualmente utiliza la Facultad de Cultura Física de la Universidad Central del Ecuador el área de bodegas.
- El sistema deberá utilizar la base de datos de la Facultad de Cultura Física de la Universidad Central del Ecuador para la solicitud de préstamos.El software deberá tener el logo de la Facultad de Cultura Física de la Universidad Central del Ecuador, el de la Universidad Central del Ecuador y los colores distintivos.

3.6. Diagrama general de los Casos de Uso

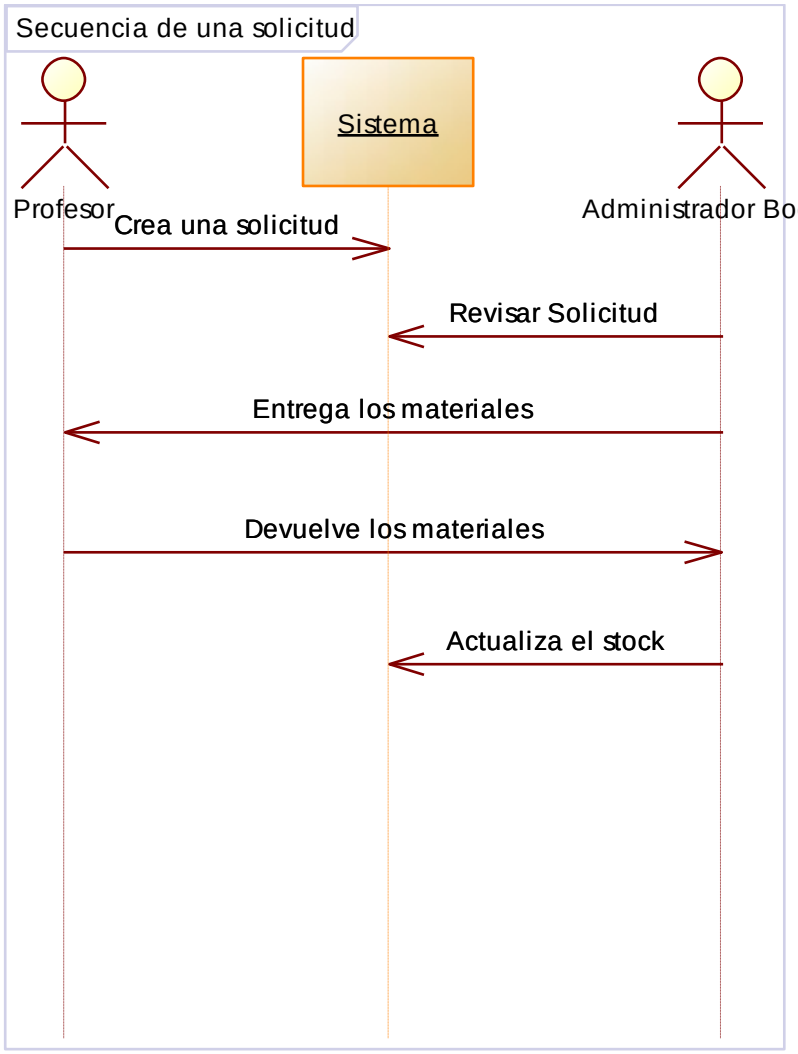


Título: Casos de uso Sistema de Inventario FCF

Autor: Tesista.

Fuente: Tesista.

3.7. Diagrama General de Secuencia



Título: Diagrama de secuencia Sistema de Inventario FCF

Autor: Tesista.

Fuente: Tesista.

El diagrama anterior describe la secuencia general del Sistema, el flujo de eventos que deben cumplirse para darse trámite a las distintas “Solicitudes de Préstamo” que se generarán. A continuación se detallara los distintos pasos:

Se crearán “Solicitudes de Préstamo” cuando un profesor de la Facultad de Cultura Física de la Universidad Central del Ecuador requiere materiales y/o implementos deportivos para dictar su clase, las cuales se notificarán al Administrador de Bodega de su creación, el mismo que,

las revisará y dará paso a la entrega de los materiales si esta fuese aceptada.

El profesor revisara los mismos y luego de su utilización entregará al administrador de bodega lo que le fue prestado.

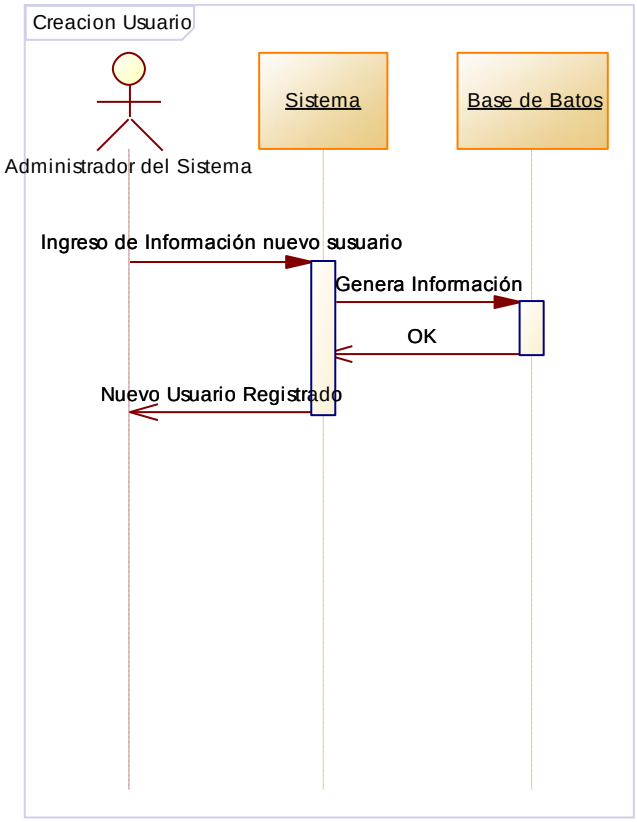
El administrador de bodega registrara la devolución de los materiales y actualizara el inventario.

3.8. Explotación de los casos de uso y sus respectivos diagramas de secuencia

3.8.1 Caso de uso: Crear Usuarios

FLUJO DE EVENTOS		
DATOS IDENTIFICATIVOS		
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos	
USUARIO RESPONSABLE:	Administrador del Sistema	
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.	
CASO DE USO:	Crear Usuarios	
FLUJO	**USUARIO**	**SISTEMA**
Básico	1. Ingresar al módulo administración de usuarios opción Ingreso. 3. Ingresar los datos del usuario. 5. Aceptar el registro del Usuario.	2. Se despliega la pantalla de nuevo registro. 4. Valida los datos Ingresados. 6. Almacenar los datos en la base de datos.
PRECONDICIONES: Haber ingresado como Administrador del Sistema		
POSTCONDICIONES:		

Diagrama de secuencia



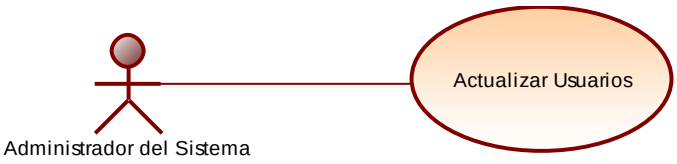
Título: Creación de Usuario

Autor: Tesista.

Fuente: Tesista.

El Administrador del Sistema puede crear usuarios.

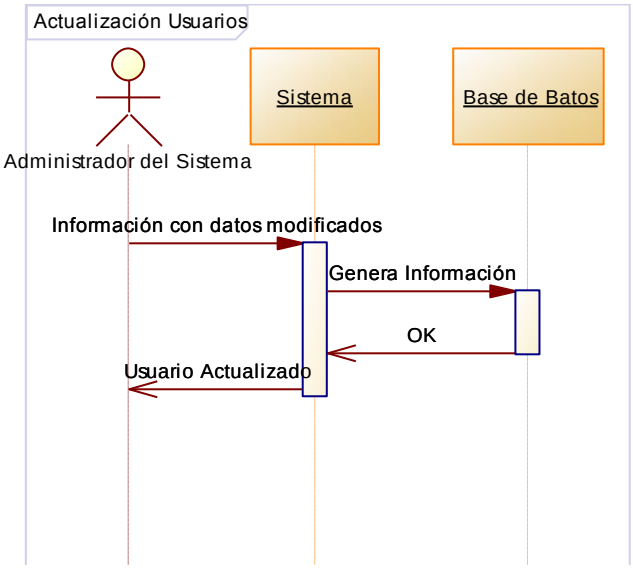
3.8.2 Caso de uso: Actualizar Usuarios



FLUJO DE EVENTOS	
DATOS IDENTIFICATIVOS	
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos
USUARIO RESPONSABLE:	Administrador del Sistema
DESCRIPCION	Es la persona responsable de establecer y mantener

RAPIDA:	el sistema.	
CASO DE USO:	Actualizar Usuarios	
FLUJO	**USUARIO**	**SISTEMA**
Básico	1. Ingresar al módulo administración de usuarios opción Ingreso. 3. Ingresar la cédula del usuario. 5. Modifica los datos del Usuario. 7. Aceptar la actualización del usuario.	2. Se despliega la pantalla de nuevo registro. 4. Valida los datos Ingresados y muestra la información del usuario. 6. Almacenar los datos en la base de datos.
PRECONDICIONES: Haber ingresado como Administrador del Sistema		
POSTCONDICIONES:		

Diagrama de secuencia



Título: Actualización de Usuarios

Autor: Tesista.

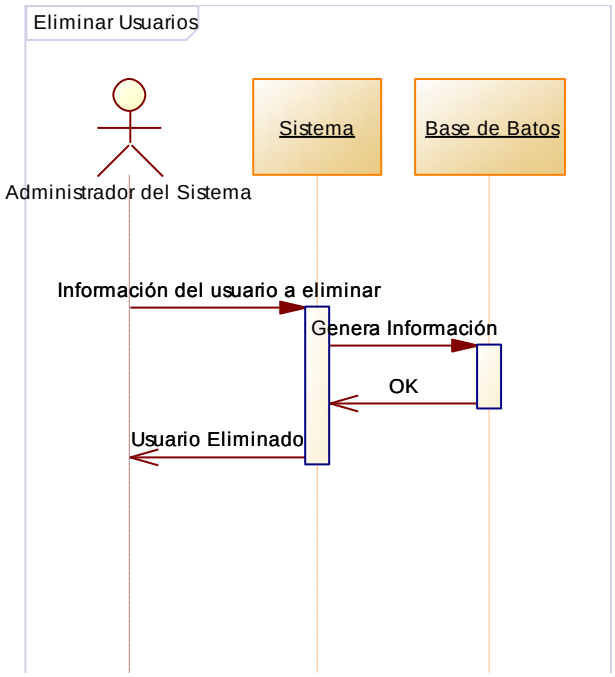
Fuente: Tesista.

El administrador puede actualizar la información de los usuarios.

3.8.3 Caso de uso: Eliminar Usuarios

FLUJO DE EVENTOS		
DATOS IDENTIFICATIVOS		
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos	
USUARIO RESPONSABLE:	Administrador del Sistema	
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.	
CASO DE USO:	Eliminar Usuarios	
FLUJO	**USUARIO**	**SISTEMA**
Básico	1. Ingresar al módulo administración de usuarios opción Ingreso. 3. Ingresar la cédula del usuario. 5. Selecciona la opción Eliminar. 7. Aceptar la eliminación del registro.	 2. Se despliega la pantalla de nuevo registro. 4. Valida los datos Ingresados y muestra la información del usuario. 6. Elimina los datos de la base de datos.
PRECONDICIONES: Haber ingresado como Administrador del Sistema		
POSTCONDICIONES:		

Diagrama de secuencia



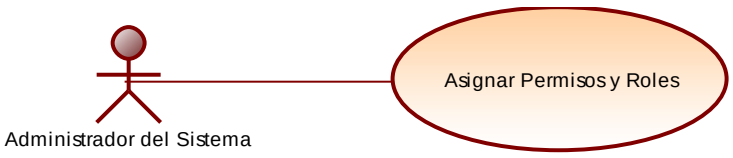
Título: Eliminación de Usuario

Autor: Tesista.

Fuente: Tesista.

El Administrador del Sistema puede eliminar usuarios, lo que se provoca un borrado lógico en la base de datos ya que se realizará un cambio de estado.

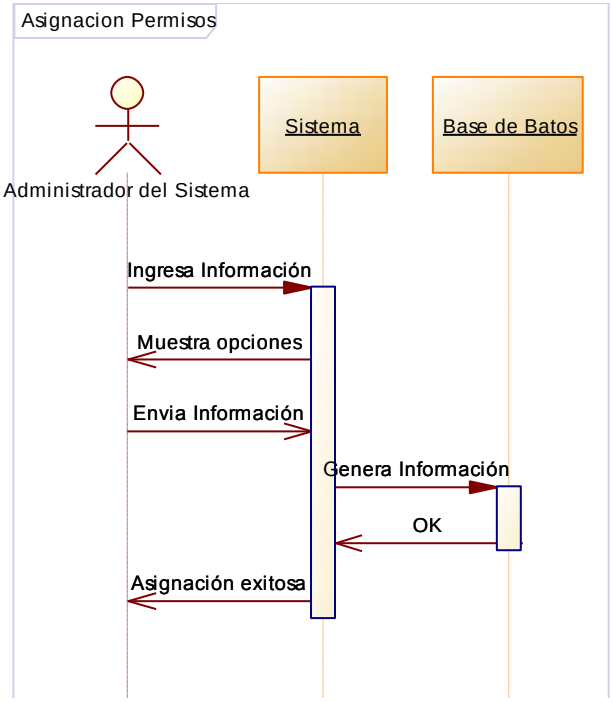
3.8.4 Caso de uso: Asignar Permisos y Roles.



FLUJO DE EVENTOS		
DATOS IDENTIFICATIVOS		
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos	
USUARIO RESPONSABLE:	Administrador del Sistema	
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.	
CASO DE USO:	Asignación de Permisos y Roles	
FLUJO	**USUARIO**	**SISTEMA**

Básico	1. Ingresar al módulo administración de usuarios opción Permisos. 3. Seleccionar el cargo y el permiso. 5. Selecciona la opción Guardar. 7. Aceptar el registro.	2. Se despliega la pantalla de nuevo registro. 4. Valida los datos Ingresados. 6. Almacena la información en la base de datos.
PRECONDICIONES: Haber ingresado como Administrador del Sistema		
POSTCONDICIONES:		

Diagrama de secuencia



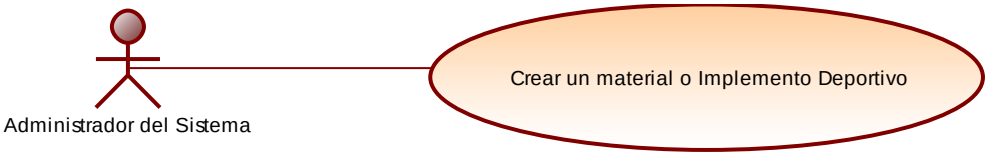
Título: Administración de permisos

Autor: Tesista.

Fuente: Tesista.

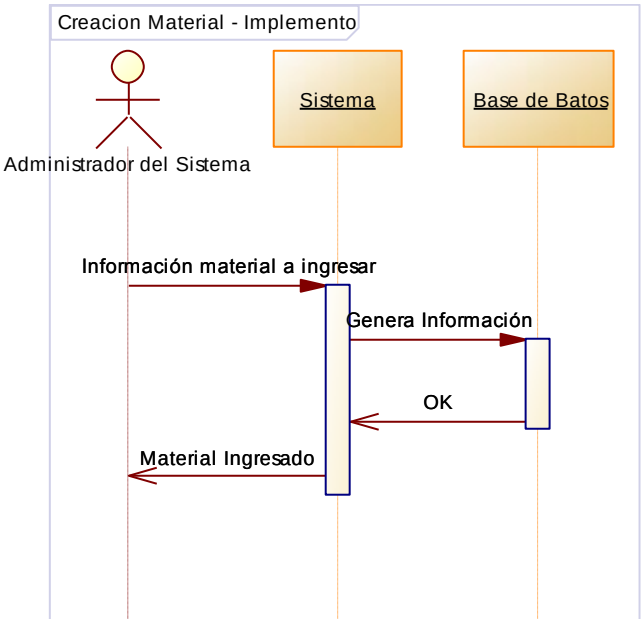
El Administrador del Sistema debe asignar los permisos correspondientes a los usuarios basado en los perfiles definidos.

3.8.5 Caso de uso: Creación de un Material o Implemento Deportivo.



FLUJO DE EVENTOS		
DATOS IDENTIFICATIVOS		
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos	
USUARIO RESPONSABLE:	Administrador del Sistema	
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.	
CASO DE USO:	Creación de un Material o Implemento Deportivo.	
FLUJO	**USUARIO**	**SISTEMA**
Básico	1. Ingresar al módulo Administración de Inventario opción Ingreso Materiales. 3. Ingresar los datos correspondientes al nuevo Material o Implemento Deportivo. 5. Selecciona la opción Guardar. 7. Aceptar el registro.	2. Se despliega la pantalla de nuevo registro. 4. Valida los datos Ingresados. 6. Almacena la información en la base de datos.
PRECONDICIONES: Haber ingresado como Administrador del Sistema		
POSTCONDICIONES:		

Diagrama de secuencia



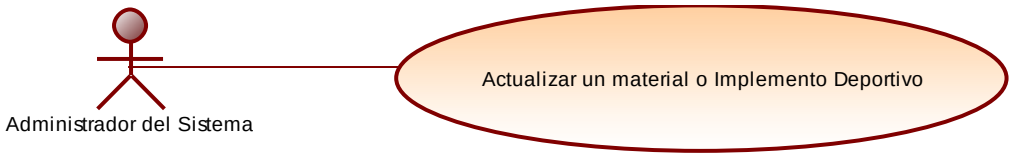
Título: Ingreso de Materiales - Implementos

Autor: Tesista.

Fuente: Tesista.

El Administrador del Sistema ingresará los materiales individualmente o por medio de la carga masiva basada en un archivo de Excel.

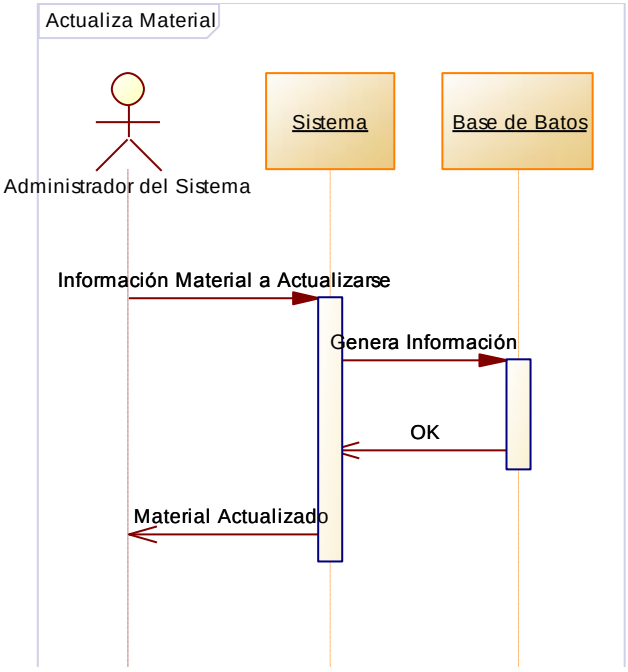
3.8.6 Caso de uso: Actualización de un Material o Implemento Deportivo.



FLUJO DE EVENTOS		
DATOS IDENTIFICATIVOS		
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos	
USUARIO RESPONSABLE:	Administrador del Sistema	
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.	
CASO DE USO:	Actualización de un Material o Implemento Deportivo.	
FLUJO	**USUARIO**	**SISTEMA**
Básico	1. Ingresar al módulo	

	Mantenimiento opción Administración Materiales. 3. Selecciona el material que desea actualizar. 5. Modifica lo que necesite y selecciona la opción Guardar. 7. Aceptar el registro.	2. Se despliega la pantalla de nuevo registro. 4. Valida los datos Ingresados. 6. Almacena la información en la base de datos.
PRECONDICIONES: Haber ingresado como Administrador del Sistema		
POSTCONDICIONES:		

Diagrama de secuencia



Título: Actualización Materiales - Implementos

Autor: Tesista.

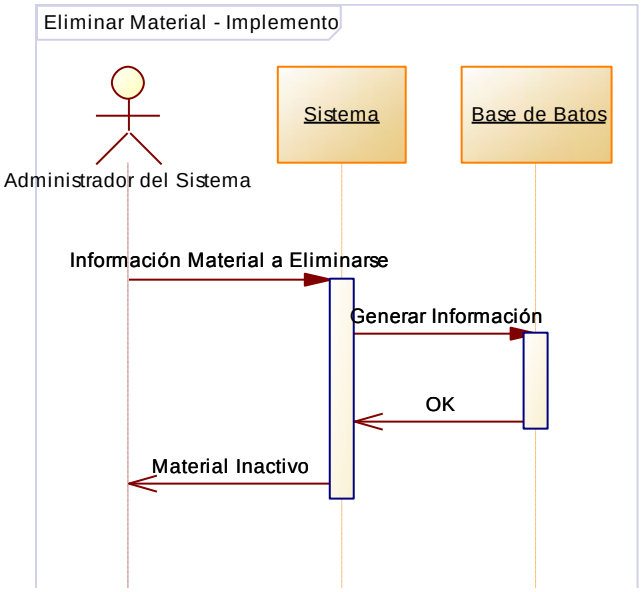
Fuente: Tesista.

El Administrador del Sistema podrá actualizar la información de los diferentes materiales ya sea en su codificación y/o su descripción.

3.8.7 Caso de uso: Eliminar un Material o Implemento Deportivo.

FLUJO DE EVENTOS		
DATOS IDENTIFICATIVOS		
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos	
USUARIO RESPONSABLE:	Administrador del Sistema	
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.	
CASO DE USO:	Actualización de un Material o Implemento Deportivo.	
FLUJO	**USUARIO**	**SISTEMA**
Básico	1. Ingresar al módulo Mantenimiento opción Administración Materiales. 3. Selecciona el material que desea eliminar. 5. Selecciona la opción Eliminar. 7. Acepta la eliminación del registro.	2. Se despliega la pantalla de nuevo registro. 4. Valida los datos Ingresados y devuelve la información. 6. Elimina la información en la base de datos.
PRECONDICIONES: Haber ingresado como Administrador del Sistema		
POSTCONDICIONES:		

Diagrama de secuencia



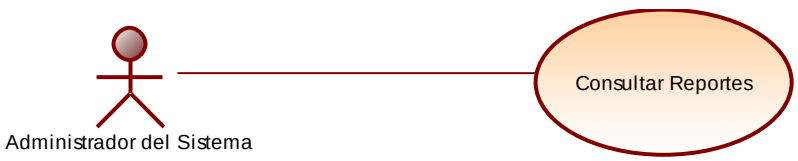
Título: Actualización Materiales - Implementos

Autor: Tesista.

Fuente: Tesista.

El Administrador del Sistema puede eliminar un material o implemento deportivo, la eliminación será lógica eso quiere decir que en la base de datos simplemente se actualizará su estado a inactivo.

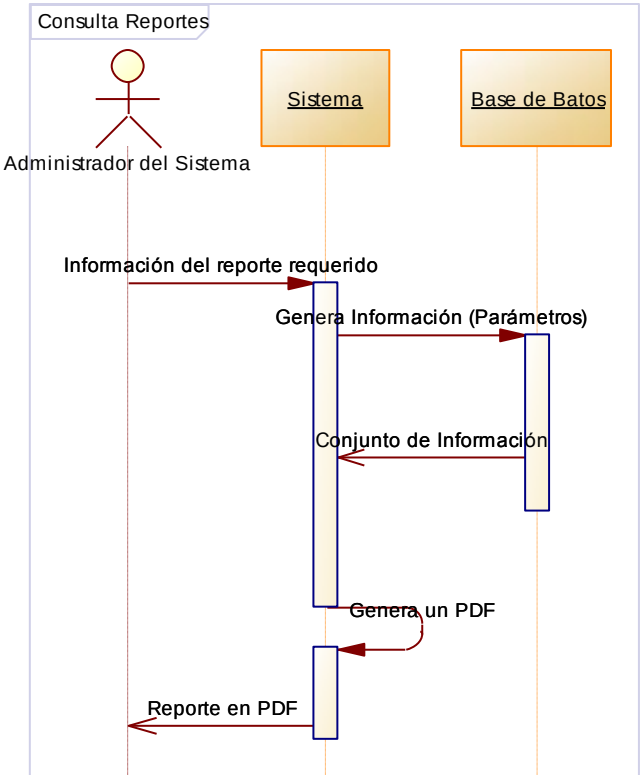
3.8.7 Caso de uso: Consultar Reportes.



FLUJO DE EVENTOS		
DATOS IDENTIFICATIVOS		
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos	
USUARIO RESPONSABLE:	Administrador del Sistema	
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.	
CASO DE USO:	Consulta de Reportes.	
FLUJO	**USUARIO**	**SISTEMA**

Básico	1. Ingresar al módulo Reportes. 3. Selecciona el reporte requerido. 5. Selecciona la opción Imprimir. 7. Verifica la información y Obtiene el informe.	2. Se despliega la pantalla de consulta de reportes de acuerdo al perfil. 4. Valida los datos Ingresados y devuelve la información. 6. Crea el reporte en formato PDF e imprime.
PRECONDICIONES: Haber ingresado como Administrador del Sistema		
POSTCONDICIONES:		

Diagrama de secuencia



Título: Reportes

Autor: Tesista.

Fuente: Tesista.

El Administrador del Sistema puede generar diferentes tipos de informes de acuerdo su necesidad.

3.8.8 Caso de uso: Consultar Reportes.

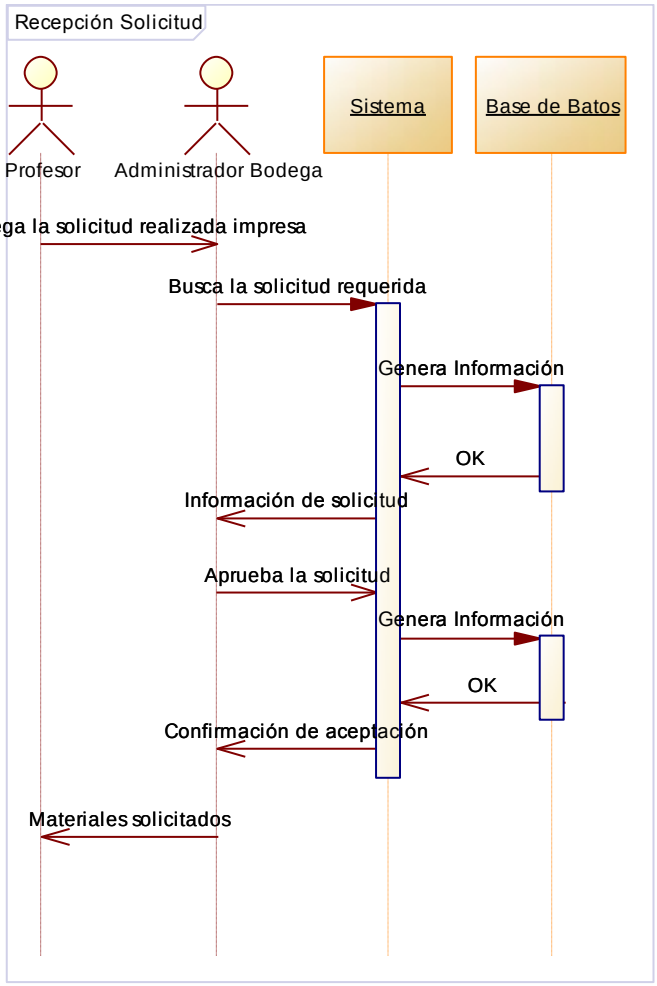
FLUJO DE EVENTOS		
DATOS IDENTIFICATIVOS		
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos	
USUARIO RESPONSABLE:	Administrador del Sistema	
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.	
CASO DE USO:	Consulta de Reportes.	
FLUJO	**USUARIO**	**SISTEMA**
Básico	1. Ingresar al módulo Reportes. 3. Selecciona el reporte requerido. 5. Selecciona la opción Imprimir. 7. Verifica la información y Obtiene el informe.	2. Se despliega la pantalla de consulta de reportes de acuerdo al perfil. 4. Valida los datos Ingresados y devuelve la información. 6. Crea el reporte en formato PDF e imprime.
PRECONDICIONES: Haber ingresado como Administrador de Bodega.		
POSTCONDICIONES:		

El Administrador de Bodega puede generar informes para gestionar su administración así como descargar información para realizar otro tipo de informes.

3.8.9 Caso de uso: Recepción de Solicitud.

FLUJO DE EVENTOS		
DATOS IDENTIFICATIVOS		
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos	
USUARIO RESPONSABLE:	Administrador del Sistema	
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.	
CASO DE USO:	Recepción de solicitudes.	
FLUJO	**USUARIO**	**SISTEMA**
Básico	1. Ingresar al módulo Solicitud de Material opción Administrar solicitud. 3. Ingresa el número de solicitud requerida. 5. Selecciona la opción Confirmar. 7. Verifica el mensaje de respuesta.	2. Se despliega la pantalla de nuevo registro. 4. Valida los datos Ingresados y devuelve la información. 6. Actualiza la información en la base de datos.
PRECONDICIONES: Haber ingresado como Administrador de Bodega.		
POSTCONDICIONES:		

Diagrama de secuencia



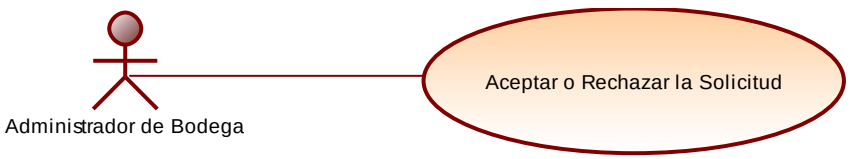
Título: Recepción Solicitud

Autor: Tesista.

Fuente: Tesista.

El Administrador de Bodega será el encargado de receptar las diferentes solicitudes y regístralas en el sistema para la actualización del stock, se generará un PDF para su respaldo.

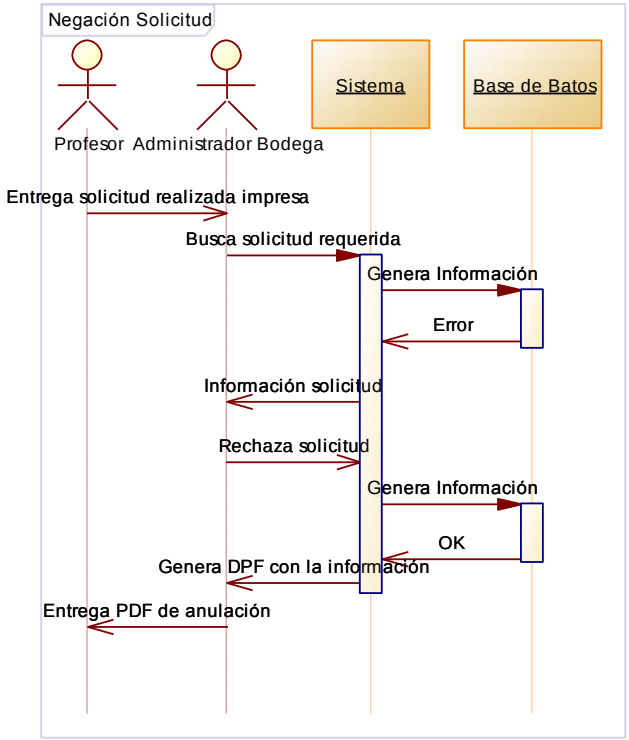
3.8.10 Caso de uso: Negación de Solicitudes.



FLUJO DE EVENTOS
DATOS IDENTIFICATIVOS

SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos	
USUARIO RESPONSABLE:	Administrador del Sistema	
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.	
CASO DE USO:	Negación de solicitudes.	
FLUJO	**USUARIO**	**SISTEMA**
Básico	1. Ingresar al módulo Solicitud de Material opción Administrar solicitud. 3. Ingresa el número de solicitud requerida. 5. Selecciona la opción Rechazar. 7. Verifica el mensaje de respuesta.	2. Se despliega la pantalla de nuevo registro. 4. Valida los datos Ingresados y devuelve la información. 6. Actualiza la información en la base de datos.
PRECONDICIONES: Haber ingresado como Administrador de Bodega.		
POSTCONDICIONES:		

Diagrama de secuencia

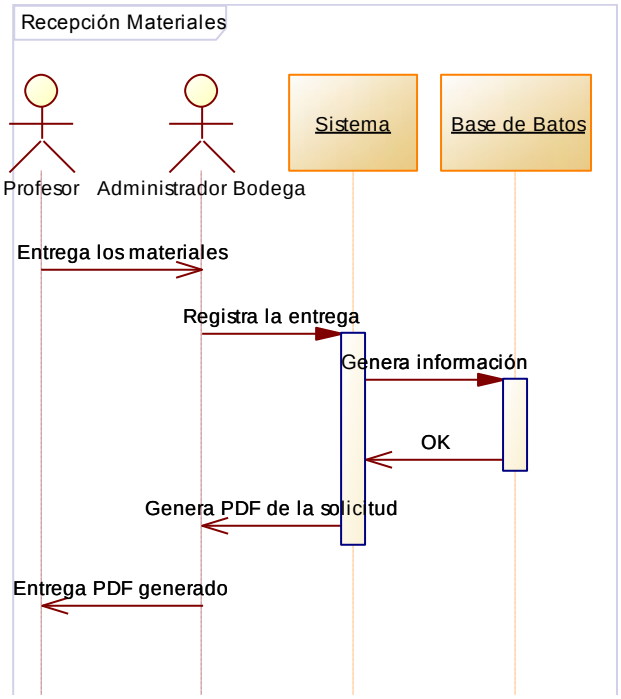


El Administrador de Bodega será el encargado de receptar las diferentes solicitudes y tendrá la potestad de rechazarlas si existe algún inconveniente con los materiales o al registrar en el sistema, se generará un PDF para su respaldo.

3.8.11 Caso de uso: Recepción de Materiales o Implementos Deportivos.

FLUJO DE EVENTOS		
DATOS IDENTIFICATIVOS		
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos	
USUARIO RESPONSABLE:	Administrador del Sistema	
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.	
CASO DE USO:	Recepción de Materiales o Implementos Deportivos.	
FLUJO	**USUARIO**	**SISTEMA**
Básico	1. Ingresar al módulo Solicitud de Material opción Administrar solicitud. 3. Ingresa el número de solicitud requerida. 5. Selecciona la opción Entregar. 7. Verifica el mensaje de respuesta y Selecciona Salir.	2. Se despliega la pantalla de nuevo registro. 4. Valida los datos Ingresados y devuelve la información. 6. Actualiza la información en la base de datos.
PRECONDICIONES: Haber ingresado como Administrador de Bodega.		
POSTCONDICIONES:		

Diagrama de secuencia



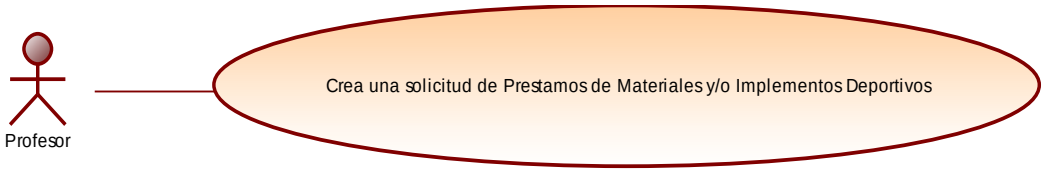
Título: Recepción Materiales

Autor: Tesista.

Fuente: Tesista.

El Administrador de Bodega será el encargado de receptar las materiales prestados por las diferentes solicitudes y regístralas en el sistema para la actualización del stock, se generará un PDF para su respaldo.

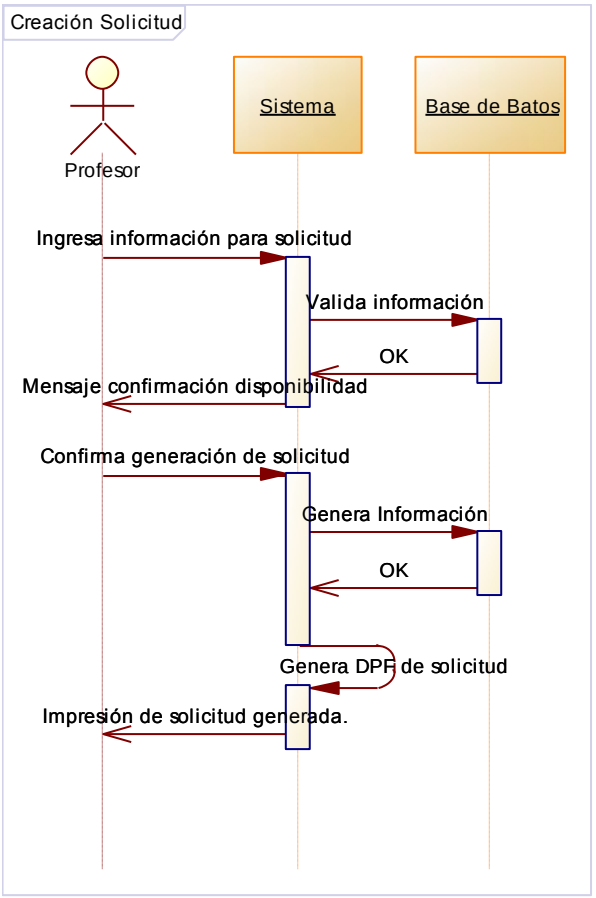
3.8.12 Caso de uso: Creación de una solicitud de Préstamo de Materiales y/o Implementos Deportivos.



FLUJO DE EVENTOS	
DATOS IDENTIFICATIVOS	
SISTEMA:	Sistema de Inventario para el Manejo de Implementos Deportivos
USUARIO RESPONSABLE:	Administrador del Sistema
DESCRIPCION RAPIDA:	Es la persona responsable de establecer y mantener el sistema.
CASO DE USO:	Recepción de Materiales o Implementos Deportivos.

FLUJO	USUARIO	SISTEMA
Básico	1. Ingresar al módulo Solicitud de Material opción Solicitud. 3. Ingresa toda la información requerida y el detalle con los materiales o implementos deportivos a ser solicitados verificando si existe el stock requerido. 5. Imprime el PDF para solicitar la entrega de materiales. 7. Verifica el mensaje de respuesta y Selecciona Salir.	2. Se despliega la pantalla de nuevo registro. 4. Valida los datos Ingresados y devuelve el número de solicitud ingresada así como un PDF de la solicitud. 6. Actualiza la información en la base de datos.
PRECONDICIONES: Haber ingresado como Profesor.		
POSTCONDICIONES:		

Diagrama de secuencia



Título: Creación Solicitud

Autor: Tesista.

Fuente: Tesista.

El Profesor como usuario normal podrá realizar la solicitud de materiales o implementos deportivos verificando su stock, se generará un PDF para su respaldo.

CAPITULO IV

4. Conclusiones y Recomendaciones

4.1 Conclusiones

Después de haber implementado el Sistema de Inventario FCF se puede llegar a las conclusiones:

- Por medio de la administración de materiales se puede ver en tiempo real la información precisa y detallada de cada uno de los materiales o implementos deportivos existentes así como su stock disponible que es fundamental para el manejo del inventario.
- Dentro de la administración de los usuarios se puede observar toda la información de cada uno de ellos y por medio de la administración de solicitudes se pude determinar que usuario y cuales materiales o implementos deportivos se están utilizando.
- Al momento de generar los reportes se obtiene información del stock de cada uno de los materiales la cual permite determinar las existencias y cuales se encuentran disponibles, en uso.
- La herramienta implementada “Sistema de Inventario FCF” es de fácil utilización ya que el ambiente es intuitivo y presta facilidades al usuario para su manejo y administración.

4.2 Recomendaciones

Como recomendaciones se pueden mencionar las siguientes:

- Se recomienda que al momento de ingresar los materiales y/o implementos deportivos al sistema se los codifique siguiendo el estándar de codificación establecido para bienes inmuebles.
- Se recomienda mantener la información actualizada de los usuarios ya que esto ayudará a llevar un control y una administración eficiente, así como un correcto funcionamiento del sistema.

- Se recomienda que el administrador del sistema realice un respaldo de la base de datos al menos una vez por semana para precautelar la información almacenada.
- Se recomienda que las claves para el ingreso al sistema se las cambie al menos una vez cada tres meses.
- Se recomienda que se desarrolle un sistema de contabilidad el cual este unificado con el sistema de inventario.
- Se recomienda que se desarrolle un sistema de seguridad para precautelar la información ya que el sistema está expuesto en la red.

BIBLIOGRAFIA

1. ZAPATA SÁNCHEZ, Pedro.(Cuarta Edición Actualizada 2002).Contabilidad General, Quito – Ecuador.

2. VÁSCONEZ ARROLLO, José Vicente.(Segunda Edición 2002). Contabilidad General Para el Siglo XXI.

3. Desarrollo Iterativo y Creciente. (Consultada: Junio 2013). Disponible en:http://www.proyectosagiles.org/desarrollo-iterativo-incremental

4. Guía de Ingeniería Del Software. (Consultada: Junio 2013). Disponible en:http://www.scribd.com/doc/62914255/Guia-de-Ingenieria-Del-Software

5. UML y Los Procesos de Desarrollo de Software. (Consultada: junio 2013). Disponible en:

http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rea_c_ji/capitulo3.pdf

6. Que es un inventario. (Consultada: julio 2013). Disponible en:

http://www.americanadeavaluos.com/index.php?option=com_content&view=article&id=5&Itemid=6

7. Definición de Control. (Consultada: Agosto 2013). Disponible en:

http://www.definicion.org/control

8. Costo de adquisición. (Consultada: Agosto 2013). Disponible en:

http://www.economicas-online.com/bienesde2.htm

9. Modelos de Inventarios. (Consultada: Octubre 2013). Disponible en:http://investigaoperativa1.blogspot.com/p/modelo-de-inventarios.html

10. Activos Fijos. (Consultada: Octubre 2013). Disponible en:

http://dspace.ups.edu.ec/bitstream/123456789/3340/1/QT01713.pdf.pdf

11. Definiciones. (Consultada: Octubre 2013). Disponible en:

http://www.definicion.org/todos/955

12. Todo Definiciones. (Consultada: Octubre 2013). Disponible en:

http://www.pkfperu.com/interpretando/bole6.doc

13. JAVA. (Consultada: Noviembre 2013). Disponible en:

http://www.ozarate.net/clases/3a_des/PRESENTACION_JAVA.pdf

14. Enciclopedia Wikipedia, Java(lenguaje de programación). (Consultada: Octubre 2013). Disponible en:http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

15. Enciclopedia Wikipedia, Java EE. (Consultada: Noviembre 2013). Disponible en:https://es.wikipedia.org/wiki/Java_EE

16. Enciclopedia Wikipedia, JavaServer Faces. (Consultada: Noviembre 2013). Disponible en:http://es.wikipedia.org/wiki/JavaServer_Faces

17. Enciclopedia Wikipedia,JBoss. (Consultada: Noviembre 2013). Disponible en:http://es.wikipedia.org/wiki/JBoss

ANEXOS

ANEXOS

Manual de Usuario

El presente documento pretende ser una guía para la utilización de la aplicación.

Una vez instalada la aplicación según se indica en el documento Manual de la aplicación accedemos a ella mediante un navegador con la dirección http://localhost:8080/SpfgmnJSF/ si se accede localmente o cambiando localhost por la dirección IP de la máquina para acceder vía remota. La siguiente pantalla es la pantalla inicial del sistema (Figura. 5).

Figura. 5 Ingreso a la aplicación

Ingreso a la Aplicación

Para ingresar al sistema el usuario debe estar previamente registrado y debe tener asignada una clave.

1. Para realizar la autentificación se debe ingresar:
    a. USUARIO (Cédula de Identidad).
    b. CLAVE.
2. Debe presionar el botón INGRESAR.

Si la autentificación esta correcta aparecerá un mensaje de bienvenida al sistema como se muestra en la Figura 6.

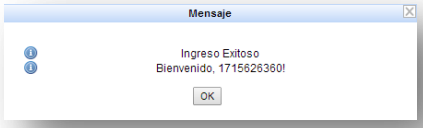


Figura. 6 Mensaje de ingreso exitoso

Una vez ingresado al sistema podemos observar el MENU PRINCIPAL como se lo puede ver en la Figura 7.

Figura. 7 Menú principal de la aplicación

Solicitud de Material

Una vez ingresada a la opción SOLICITUD DE MATERIAL tenemos un submenú, en él se muestra las opciones de creación de una solicitud (SOLICITUD) y la administración de una solicitud (ADMINISTRAR SOLICITUD). (Figura 8)

Figura. 8 Menú solicitud de material

Creación de una solicitud

Para crear una solicitud se debe seguir los siguientes pasos: (Figura 9 y Figura 10)

1. Ingrese a la opción Solicitud de Material.
2. Ingrese a la opción Solicitud.

3. Ingrese la cédula.
4. Presione la lupa ubicada a la derecha de la casilla cédula.
5. Ingrese la duración en horas.
6. Ingrese el curso
7. Seleccione el tipo de material.
8. Seleccione el material.
9. Si lo requiere revise el stock.
10. Ingrese la cantidad.
11. De clic en el botón Agregar.
12. Repita el proceso hasta agregar todos los materiales requeridos.
13. Presione el botón Guardar.

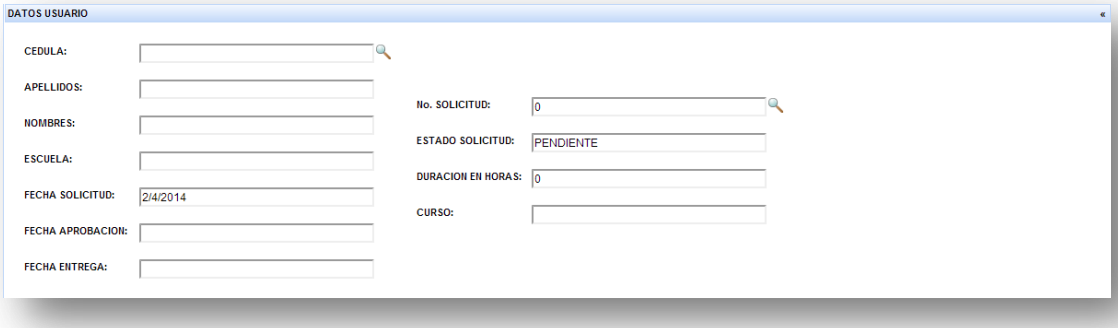


Figura. 9 Cabecera de la solicitud

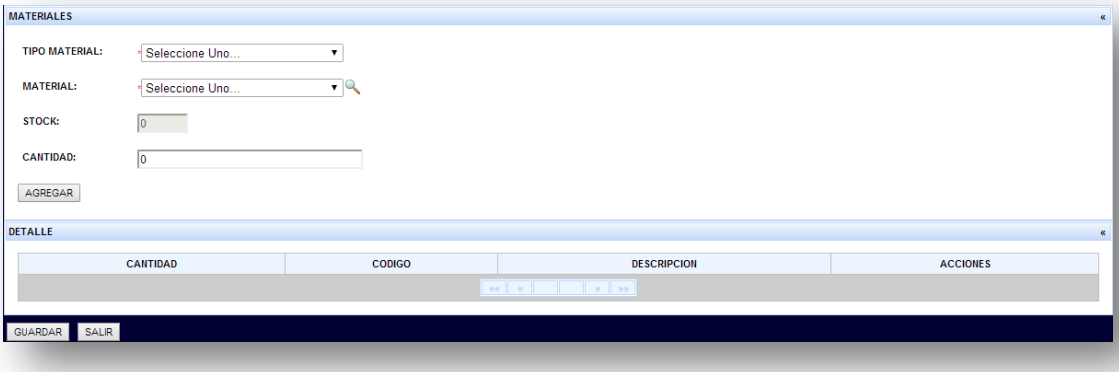


Figura. 10 Detalle de la solicitud

Administrar una solicitud

Dentro de la Administración de una solicitud tenemos varias opciones las cuales se listan a continuación y esto dependerá del estado de la solicitud:

1. Confirmar una solicitud.

2. Rechazar una solicitud.
3. Entregar una solicitud.

En la Figura 11 se puede apreciar la pantalla principal de la administración de una solicitud y los botones para confirmar, rechazar y entregar se harán visibles de acuerdo al estado de la solitud.

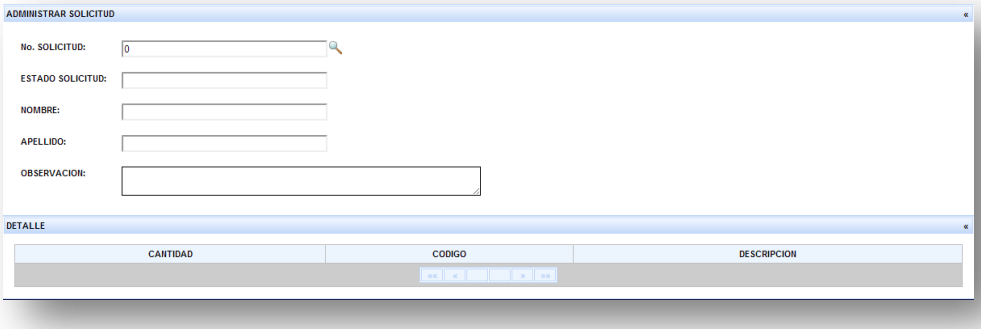


Figura. 11 Administración de solicitud

Confirmar una Solitud:

Para confirmar una solicitud se deben seguir los siguientes pasos:

1. Ingrese a la opción Solicitud de Material.
2. Ingrese a la opción Administración de Solicitud.
3. Ingrese el número de solicitud.
4. De clic en la lupa ubica a la derecha de la casilla No. Solicitud.
5. Ingrese una observación si es necesario.
6. De clic en el botón Confirmar.

Rechazar una Solitud:

Para rechazar una solicitud se deben seguir los siguientes pasos:

1. Ingrese a la opción Solicitud de Material.
2. Ingrese a la opción Administración de Solicitud.
3. Ingrese el número de solicitud.
4. De clic en la lupa ubica a la derecha de la casilla No. Solicitud.
5. Ingrese una observación si es necesario.
6. De clic en el botón Rechazar.

Entregar una Solitud:

Para entregar una solicitud se deben seguir los siguientes pasos:

1. Ingrese a la opción Solicitud de Material.
2. Ingrese a la opción Administración de Solicitud.
3. Ingrese el número de solicitud.
4. De clic en la lupa ubica a la derecha de la casilla No. Solicitud.
5. Ingrese una observación si es necesario.
6. De clic en el botón Entregar.

Administración de materiales

Para realizar la administración de materiales tenemos tres partes las cuales son:

1. Ingreso de Materiales Individuales.
2. Ingreso de Stock de Materiales.
3. Carga Masiva de Materiales.

Ingreso de Materiales

Para realizar el ingreso de materiales individuales se debe realizar el siguiente procedimiento (Figura 12).

1. Seleccione el tipo de material.
2. Ingrese la descripción.
3. Ingreso el código
4. Ingrese las existencias mínimas.
5. De clic en el botón Guardar correspondiente al panel

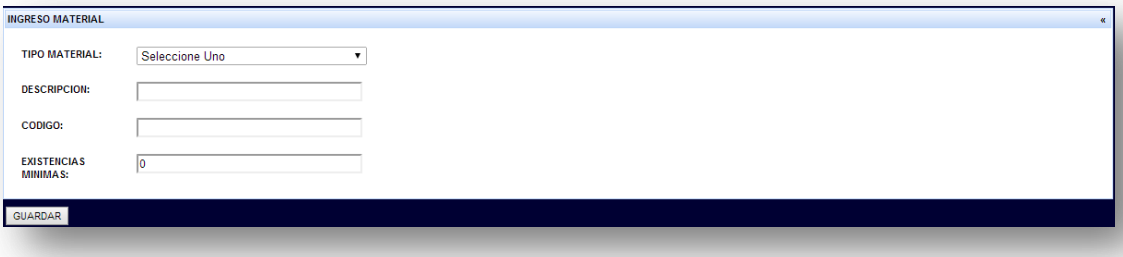


Figura. 12 Ingreso de material

Ingreso de Stock de Materiales

Para realizar el ingreso de stock de materiales se debe primero hacer el registro del material individualmente y de ahí se realiza siguiente procedimiento (Figura 13).

1. Seleccione el material.

2. Ingrese la cantidad
3. De clic en el botón Guardar correspondiente al panel.

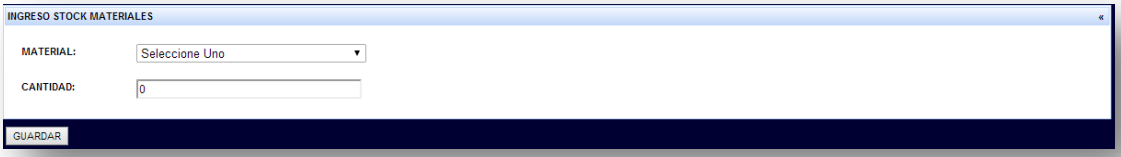


Figura. 13 Ingreso stock de materiales

Carga Masiva de Materiales

La carga masiva de materiales nos permite realizar el ingreso de materiales por medio de un archivo con formato .xls, para ello se debe seguir el siguiente procedimiento (Figura 14).

1. Ubíquese en el panel CARGA MATERIALES ARCHIVO.
2. De clic en Agregar Archivo.
3. Seleccione el archivo correspondiente mediante el explorador.
4. De clic en el botón Ejecutar.

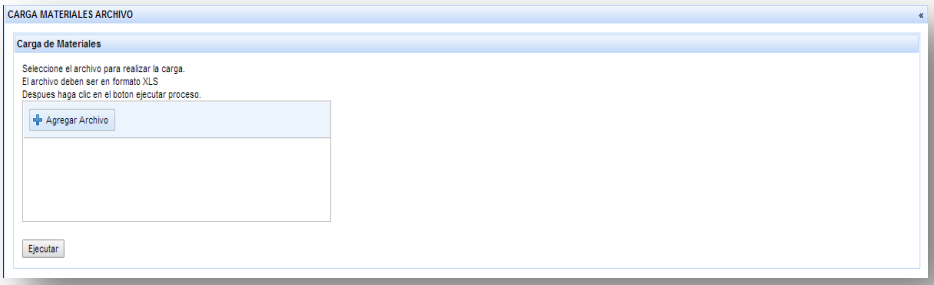


Figura. 14 Carga masiva de materiales

Reportes

Dentro de esta opción se van a generar reportes de acuerdo a la búsqueda seleccionada, con la ayuda de las tres alternativas posibles y estas son:

1. Material.
2. Código.
3. Tipo Material.

El procedimiento sería el siguiente (Figura 15)

1. Seleccione la opción por la cual quiere realizar el reporte.
    a. Material
    b. Código.
    c. Tipo Material.
2. De clic en la lupa correspondiente a la opción seleccionada.
3. Descargarse a Excel si es necesario.
4. Descargarse a PDF si es necesario.

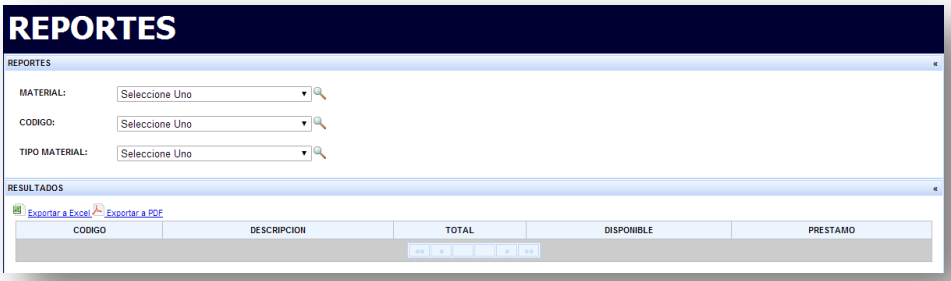


Figura. 15 Generación de reportes

Administración de usuarios

En la opción Administración de Usuarios se realizará el ingreso de las personas que van a utilizar el sistema y en esta pantalla se podrá realizar su administración como es el de buscar, crear, actualizar y eliminar un usuario (Figura 16).

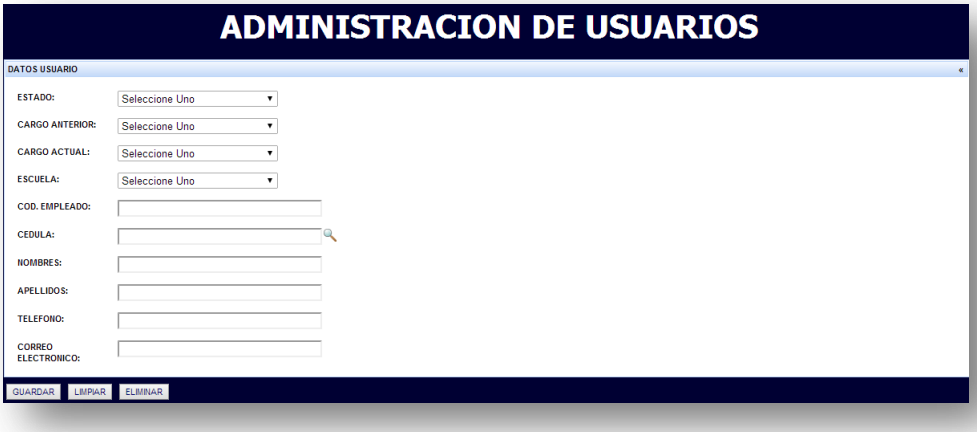


Figura. 16 Administración de usuarios

Registro de un Usuario.

1. Ingrese a la opción Administración de Usuarios

2. Seleccione el Estado
3. Seleccione el Cargo Anterior
4. Seleccione el Cargo Actual
5. Seleccione la Escuela a la que pertenece.
6. Ingrese el Código de empleado.
7. Ingrese la Cédula.
8. Ingrese los Nombres.
9. Ingrese los Apellidos.
10. Ingrese el Teléfono.
11. Ingrese el Correo Electrónico.
12. De clic en el botón Guardar.

Búsqueda de un Usuario

1. Ingrese a la opción Administración de Usuarios
2. Ingrese el número de Cédula.
3. Seleccione buscar (Presionar la lupa ubicada en la parte derecha de la casilla correspondiente a la cédula).

Actualización de un Usuario

1. Ingrese a la opción Administración de Usuarios
2. Realice la búsqueda de usuario.
3. Realice los cambios requeridos.
4. De clic en el botón Guardar.

Eliminación de un Usuario

1. Ingrese a la opción Administración de Usuarios
2. Realice la búsqueda de usuario.
3. De clic en el botón Eliminar.

Mantenimiento

Dentro de la opción Mantenimiento tenemos la administración de las tablas que son listas de valores así como la administración de permisos y asignación de claves (Figura 17).

Figura. 17 Menú mantenimiento

Catálogo

Por medio de esta opción se realizará la administración de la cabecera para las diferentes listas de valores, dentro la cual tenemos estado, cargo, escuela, entre otras (Figura 18).

Para poder listar los datos ya ingresados se debe presionar Buscar.

Esta pantalla está diseñada solo para el ingreso y búsqueda de registros, la modificación se la realizará a nivel de base de datos por el administrador del sistema esto debido a que la eliminación o edición de los campos registrados afectaría el normal funcionamiento del sistema.

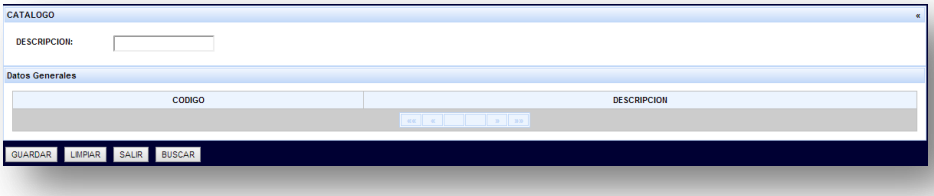


Figura. 18 Cabecera catálogo

Detalle Catálogo

Dentro de la opción Detalle catálogo se realizará la administración del detalle de las diferentes listas de valores correspondientes al Catálogo (Figura 19).

Para el ingreso se deberá seleccionar la cabecera e ingresar la descripción del detalle y presionar el botón guardar.

Al igual que la cabecera el sistema está diseñado para el ingreso de registro, la actualización o eliminación la realizara el administrador del sistema por medio de la base de datos.

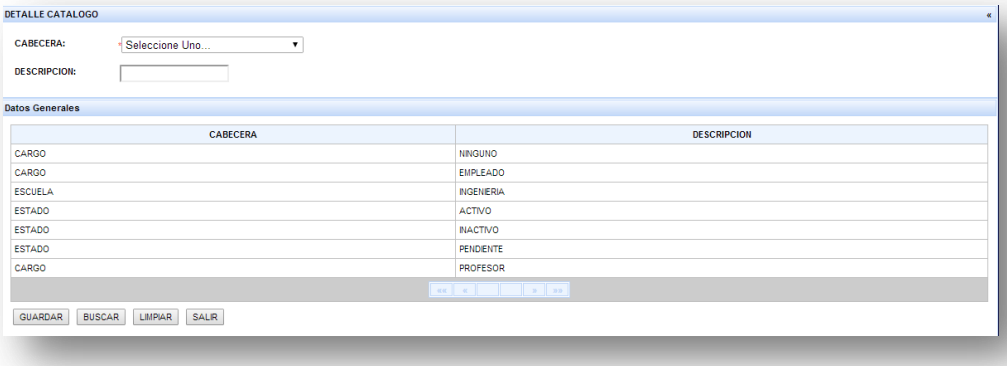


Figura. 19 Detalle catálogo

Ingreso Tipo de Material

Por medio de esta opción se realizará la administración de los tipos de materiales, esto hace referencia a la diferenciación que habrá entre los materiales e implementos deportivos así como los utilizados en suficiencia o en materias de carrera (Figura 20).

Para registrar un nuevo tipo se ingresará su descripción en la casilla correspondiente y se presionará el botón Guardar.

Por medio del botón Listar se podrán ver los registros ingresados.

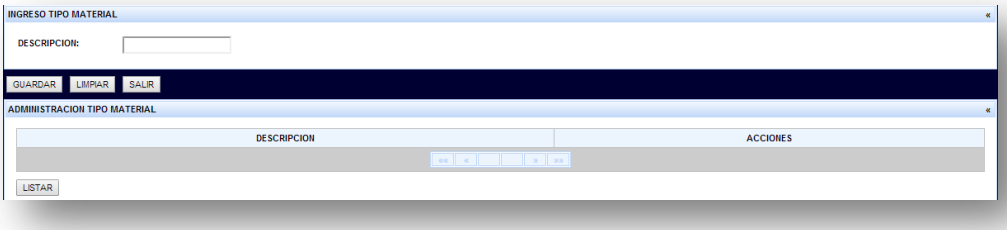


Figura. 20 Ingreso tipo de material

Administración de claves

Ingreso de claves

Para Ingresar una clave se debe seguir el siguiente procedimiento (Figura 21).

1. Ingrese a la opción Mantenimiento.
2. Ingrese a la opción Ingreso Claves.
3. Ubicar la opción Ingreso Clave.
4. Ingrese la Cédula.

5. Ingrese la Clave.
6. Confirme la Clave.
7. De clic en el botón Guardar.

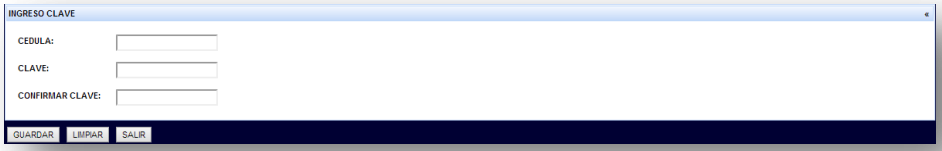


Figura. 21 Ingreso clave

Cambio de claves

Para cambiar una clave se debe seguir el siguiente procedimiento (Figura 22).

1. Ingrese a la opción Mantenimiento.
2. Ingrese a la opción Ingreso Claves.
3. Ubicar la opción Cambio Clave.
4. Ingrese la Cédula.
5. Ingrese la Clave Anterior.
6. Ingrese la Clave Nueva.
7. Confirme la Clave.
8. De clic en el botón Guardar.

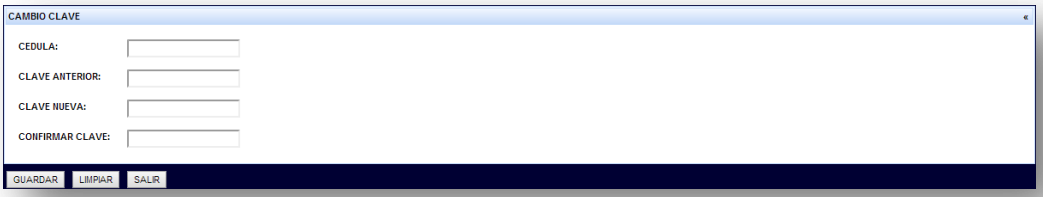


Figura. 22 Cambio de clave

Reseteo claves

Para resetear una clave se debe seguir el siguiente procedimiento (Figura 23).

1. Ingrese a la opción Mantenimiento.
2. Ingrese a la opción Ingreso Claves.
3. Ubicar la opción Resetear Clave.

4. Ingrese la cédula.
5. De clic en el botón Resetear. (UCE clave por defecto.)

Figura. 23 Reseteo de clave

Administración de permisos

En la opción administración de permisos re registrarán todos los permisos que existen en el sistema tanto como los de ingreso así como los de acción.

Para registrar un nuevo permiso se debe ingresar su código, su descripción y presionar el botón guardar.

Los permisos se podrán editar por medio de la opción Editar ubicada a la altura de cada permiso dentro de la tabla de despliegue, los registros de los permisos no podrán ser eliminados ya que esto afectaría el normal funcionamiento del sistema (Figura 24).

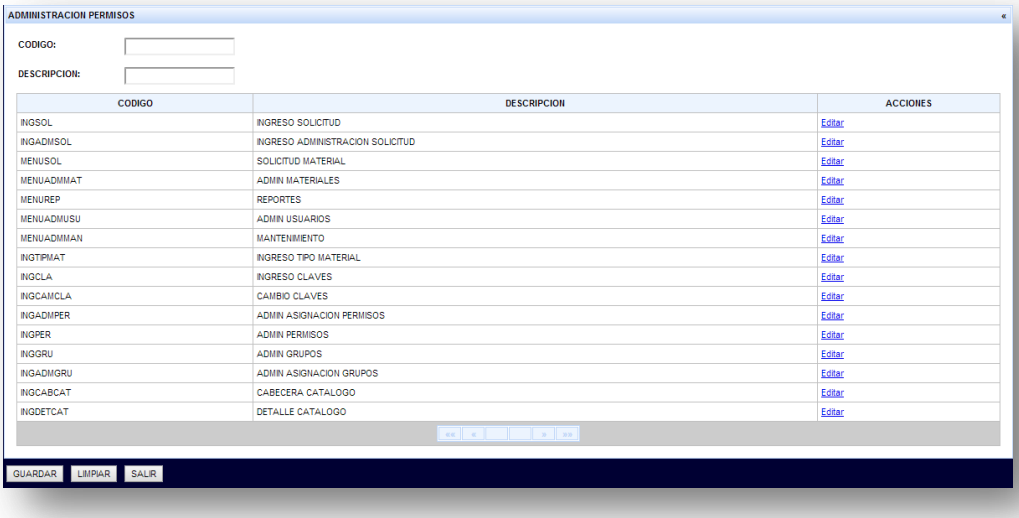


Figura. 24 Administración de permisos

Administración de asignación de permisos

En la opción administración de permisos se ligarán los permisos existentes a los diferentes grupos.

Para asignar un permiso a un grupo determinado se deberá seleccionar el grupo y el permiso correspondiente para luego presionar el botón de guardar.

Los permisos asignados podrán ser editados por medio de la opción Editar que se encuentra a la altura de cada registro así como pueden ser deshabilitados pasándolos a un estado inactivo por medio de la opción Cambiar estado ubicada a la altura de cada registro (Figura 25).

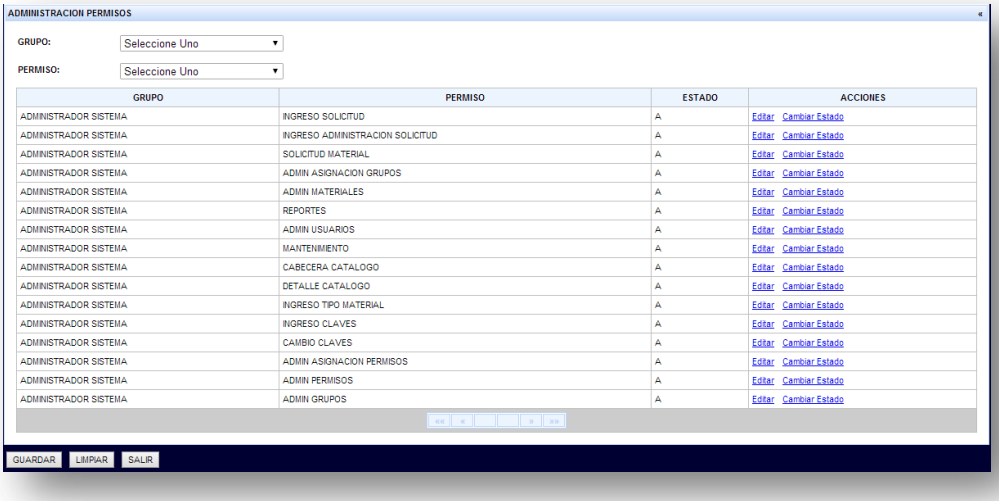


Figura. 25 Administración de asignación de permisos

Administración de grupos

Por medio de la opción Administración de Grupos se podrá ingresar nuevos grupos.

Para ingresar se deberá ingresar la descripción del grupo y presionar la opción Guardar.

Los grupos ingresados no podrán ser editados o eliminados ya que esto afectará el funcionamiento normal del sistema (Figura 26).

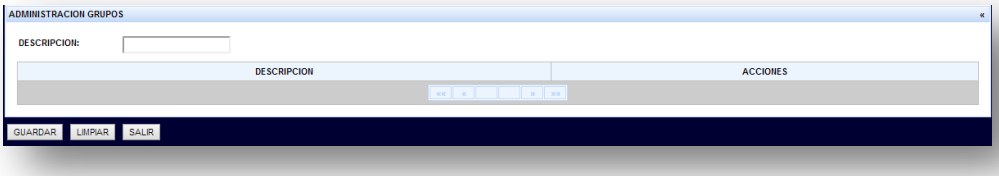


Figura. 26 Administración de grupos

Administración asignación de grupos

En la opción administración asignación de grupos se ligarán los usuarios existentes a los diferentes grupos ingresados para poder formar los perfiles.

Para asignar un usuario a un grupo determinado se deberá seleccionar el grupo y el usuario correspondiente para luego presionar el botón de guardar.

Los grupos asignados podrán ser editados por medio de la opción Editar que se encuentra a la altura de cada registro.

Los grupos asignados podrán ser eliminados por medio de la opción Eliminar ubicada a la altura de cada registro (Figura 27).

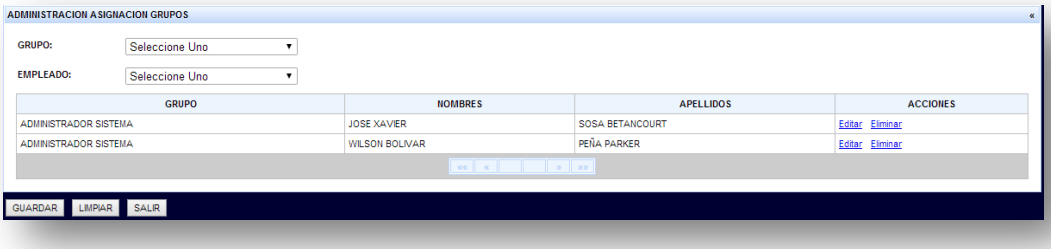


Figura. 27 Administración de asignación de grupos

MANUAL DE INSTACIÓN AMBIENTE DE DESARROLLO

Instalación del JDK

Copiar el instalador jdk-6u45-linux-i586.bin en la carpeta /opt.

Abrir un Terminal como administrador (root).

Ingresar a la carpeta opt.

Escribir la ubicación del archivo copiado (/opt/jdk-6u45-linux-i586.bin) y presionar enter.

Esperar hasta que el instalador termine de ejecutarse.

Establecer la variable de entorno JAVA_HOME y modificar la variable PATH.

Ubicarse en el fichero /etc/profile.

Ubicarse al final del archivo y escribir:

```
export JAVA_HOME=/opt/jdk1.6.0_45
export PATH=$PATH:$JAVA_HOME/bin/
```

Cambiar la versión de Java del sistema operativo

Escribir:

```
[root@localhost opt]# alternatives --install /usr/bin/java java /opt/jdk1.6.0_45/bin/java 100
[root@localhost opt]# alternatives --install /usr/bin/javac javac /opt/jdk1.6.0_45/bin/javac 100
[root@localhost opt]# alternatives --install /usr/bin/jar jar /opt/jdk1.6.0_45/bin/jar 100
[root@localhost opt]# alternatives --config java
```

Muestra las versiones de java existentes

```
  Selección    Comando
-----------------------------------------------
*+ 1           /usr/lib/jvm/jre-1.7.0-openjdk/bin/java
   2           /usr/lib/jvm/jre-1.6.0-openjdk/bin/java
   3           /opt/jdk1.6.0_45/bin/java
```

Seleccionar la versión instalada (3).

Para verificar la versión instalada escribimos: java –version.

```
[root@localhost opt]# java -version
java version "1.6.0_45"
Java(TM) SE Runtime Environment (build 1.6.0_45-b06)
Java HotSpot(TM) Client VM (build 20.45-b01, mixed mode, sharing)
```

Instalación de PostgreSQL 9.3

Copiar el instalador postgresql-9.3.2-1-linux.run en la carpeta /opt.

Abrir un Terminal como administrador (root).

Ingresar a la carpeta opt.

Escribir la ubicación del archivo copiado (/opt/postgresql-9.3.2-1-linux.run) y presionar enter.

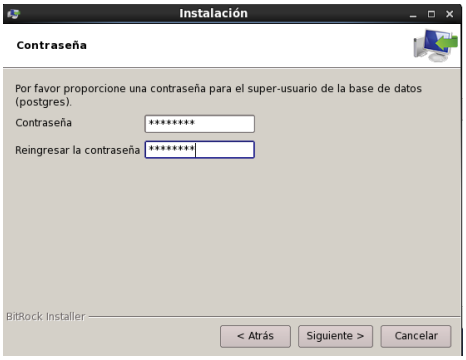
```
[root@localhost ~]# cd /opt
[root@localhost opt]# postgresql-9.3.2-1-linux
bash: postgresql-9.3.2-1-linux: no se encontró la orden
[root@localhost opt]# ls
jdk1.6.0_45  jdk-6u45-linux-i586.bin  postgresql-9.3.2-1-linux.run  rh
[root@localhost opt]# postgresql-9.3.2-1-linux.run
bash: postgresql-9.3.2-1-linux.run: no se encontró la orden
[root@localhost opt]# '/opt/postgresql-9.3.2-1-linux.run'
```

Iniciará el wizard de la instalación y presionamos Siguiente.

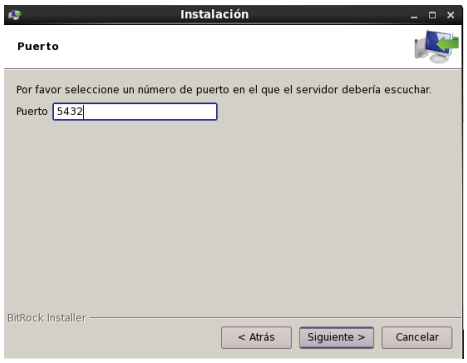
Ingresamos el directorio donde se va a realizar la instalación (para nuestro caso dejamos el mismo que nos indica).

Ingresamos el directorio donde se van a almacenar los datos (para nuestro caso dejamos el mismo que nos indica).

Ingresamos la clave para el administrador (para nuestro caso ingresaremos **postgres**).

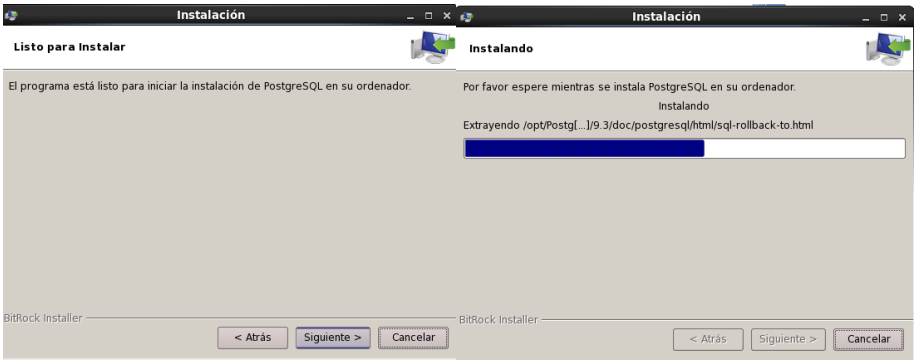


Ingresamos el puerto por el cual se conectaran al servidor (5432).



Ingresamos la configuración regional (para nuestro caso dejaremos la misma que nos indica).

En esta pantalla nos indica que la instalación dará inicio.



En la siguiente pantalla nos indica que la instalación a terminado, quitamos el check del Stack Builder y presionamos Terminar.

Configuración JBOSS como servicio

Ubicar el archivo jboss_init_redhat.sh en la ruta: /opt/jboss-5.1.0.GA/bin

Abrirlo con el editor de texto.

Luego de la primera línea escribir: (Ver Gráfico 1).

chkconfig: 345 91 10

description: Star or stop SRVJBOSS

Cambiar las rutas de los directorios correspondientes al jboos y jdk deacuerdo a la configuración que se tenga (Figura 28.).

```
#!/bin/sh
# chkconfig: 345 91 10
# description: Star or stop SRVJBOSS
#
# $Id: jboss_init_redhat.sh 81068 2008-11-14 15:14:35Z dimitris@jboss.org $
#
# JBoss Control Script
#
# To use this script run it as root - it will switch to the specified user
#
# Here is a little (and extremely primitive) startup/shutdown script
# for RedHat systems. It assumes that JBoss lives in /usr/local/jboss,
# it's run by user 'jboss' and JDK binaries are in /usr/local/jdk/bin.
# All this can be changed in the script itself.
#
# Either modify this script for your requirements or just ensure that
# the following variables are set correctly before calling the script.

#define where jboss is - this is the directory containing directories log, bin, conf etc
JBOSS_HOME=${JBOSS_HOME:-"/opt/jboss-5.1.0.GA/"}

#define the user under which jboss will run, or use 'RUNASIS' to run as the current user
JBOSS_USER=${JBOSS_USER:-"RUNASIS"}

#make sure java is in your path
JAVAPTH=${JAVAPTH:-"/opt/jdk1.6.0_45/bin"}

#configuration to use, usually one of 'minimal', 'default', 'all'
JBOSS_CONF=${JBOSS_CONF:-"default"}

#if JBOSS_HOST specified, use -b to bind jboss services to that address
JBOSS_BIND_ADDR=${JBOSS_HOST:+"-b $JBOSS_HOST"}
```

Figura. 28 Configuración ruta de directorios

Abrir un Terminal como administrador.

Ingresar a la ruta /etc/init.d

Escribir:

ln –s /opt/jboss-5.1.0.GA/bin/jboss_init_redhat.sh SRVJBOSS

chkconfig –add SRVJBOSS

```
[root@localhost ~]# cd /etc/init.d
[root@localhost init.d]# ln -s /opt/jboss-5.1.0.GA/bin/jboss_init_redhat.sh SRVJBOSS
[root@localhost init.d]# chkconfig --add SRVJBOSS
[root@localhost init.d]#
```

Para levantar el servicio ingresamos:

service SRVJBOSS star

Para para el servicio ingresamos:

service SRVJBOSS stop

MANUAL TÉCNICO

Introducción

Este documento provee la información técnica necesaria para el mantenimiento y administración del **SISTEMA DE INVENTARIO PARA EL MANEJO DE IMPLEMENTOS DEPORTIVOS**.

Este manual ayuda a entender como está desarrollado e implementado el sistema en caso de mantenimiento y cambios de funcionalidad del mismo.

Objetivos

Detallar cada una de las clases empleadas para el desarrollo del proyecto en las diferentes herramientas libres:

- Eclipse
- JBoss 5.1
- PostgreSQL-9.x
- JDK6.u45.

Herramientas utilizadas

ECLIPSE

Herramienta de programación de java.

JBOSS 5.x

Servidor de Aplicaciones.

POSTGRES 9.x

Base de datos.

JDK 6.u46

Máquina Virtual.

Estándares de programación

Los nombres de los PAQUETES estarán dados de acuerdo a la funcionalidad y al módulo de desarrollo al que corresponden.

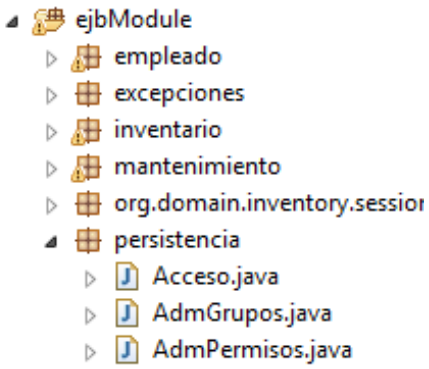
Los nombres de las CLASES estarán ligadas al funcionamiento de cada una de ellas.

Todo el mapeo de las tablas estarán dentro de un solo paquete llamada Persistencia.

Los nombres de las páginas describirán las acciones que van a realizar y estas estarán ligadas a sus respectivos Beans.

JPA (Java Persistence API)

Más conocida por su sigla JPA, es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API nos ayuda con el mapeo objeto-relación. El objetivo de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos



Dentro del paquete persistencia, se encuentran todas las clases que nos permiten manejar la persistencia a continuación describiremos una como ejemplo:

Acceso.java

Las librerías usadas para generar la persistencia son:

```
import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;
```

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.SequenceGenerator;

import javax.persistence.Table;

import org.hibernate.envers.AuditTable;

import org.hibernate.envers.Audited;

import auditoria.ConstantesAuditoria;

Las anotaciones utilizadas son las siguientes:

@Entity: Nos indica que es una entidad

@Table(name="acceso", schema = "inventario"): Nos indica con que tabla de la base de datos está relacionada así como su esquema.

@Audited: Nos indica que va a estar auditado por medio de una librería llamada Envers.

@AuditTable(value = "aut_acceso", schema = ConstantesAuditoria.SCHEMA): Nos indica con que tabla de auditoria está ligada así como su esquema.

@Id: Nos indica cual es la clave foránea.

@SequenceGenerator(name="id_acceso", sequenceName = "inventario.id_acceso_seq"): Nos indica que campo de la tabla en la base de datos va hacer el secuencial.

@GeneratedValue(strategy=GenerationType.SEQUENCE, generator="id_acceso"): Nos indica que la generación del secuencia lo va a manejar la entidad.

@Column(name="id_acceso"): Nos indica cómo se llama la columna relacionada en la base de datos así como sus propiedades.

@ManyToOne(fetch=FetchType.EAGER): Nos indica cómo y cuál es la relacion que existe con la tabla relacionada para este caso id_empleado.

@JoinColumn(name="id_empleado"): Nos indica con cual columna de la otra tabla se está haciendo la relacion.

EJB

Los EJB o Enterprise Java Beans son componentes JEE que se ejecutan dentro de un container EJB, en un entorno de ejecución dentro de un Application

Server. El contenedor de EJB provee servicios al usuario, los cuales expone de manera transparente,estos servicios pueden ser Transacciones, Seguridad, etc. Los Entreprise Java Beans son componentes del lado del servidor que encapsulan la lógica del negocio de una aplicación.

Los EJB encapsulan operaciones accesibles de modo remoto desde los clientes finales o desde los componentes de la capa WEB, denominada capa de presentación.

Para este proyecto utilizaremos tipos de EJBs específicos que se detallan a continuación.

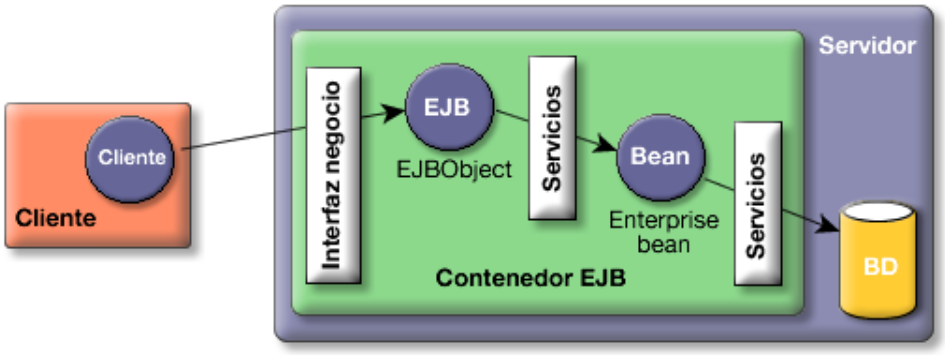
Conversation EJB.-Este caracteriza por implementar operaciones síncronas (llamada-respuesta) y son de varios tipos entre los cuales tenemos:

@Stateful: Mantienen el estado conversacional con el cliente, es decir, mantiene los atributos y sus valores entre llamadas mientras el cliente invocador este activo.

@Stateless: No mantienen estado (son independientes de los clientes)

También se utilizará un EJB de tipo:

@Remote: Que es un nuevo tipo de solución para servicios que utiliza una sencilla herramienta de conexión para recopilar y recibir datos.

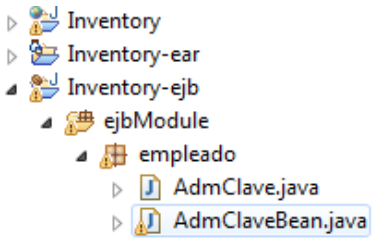


Tecnología EJB

Fuente: http://www.jtech.ua.es/j2ee/2003-2004/abierto-j2ee-2003-2004/ejb/sesion01-apuntes.htm

EL proyecto está compuesto por paquetes que contienen a los diferentes módulos dentro los cuales encontramos las interfaces que las podemos identificar por que llevan la notación @Local y estas están ligadas a los beans de implementación que serán de tipo @Stateful que son los que

manejarán sus interfaces correspondientes y dentro de los cuales estará toda la lógica del comportamiento de los métodos.



A continuación se detallara un ejemplo:

AdmClaveBean.java

Es la implementación que contiene toda la lógica de funcionalidad para la administración del cambio de claves y esta está ligada a su interfaz AdmClave.java a su página admClave.xhtml que se describirán luego.

package empleado;

- Nos indica dentro de que paquete se encuentra el bean.

```
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.EJB;
import javax.ejb.Remove;
import javax.ejb.Stateful;
import org.jboss.seam.ScopeType;
import org.jboss.seam.annotations.Begin;
import org.jboss.seam.annotations.End;
import org.jboss.seam.annotations.In;
import org.jboss.seam.annotations.Logger;
import org.jboss.seam.annotations.Name;
import org.jboss.seam.annotations.Scope;
import org.jboss.seam.faces.FacesMessages;
import org.jboss.seam.log.Log;
import persistencia.Acceso;
import persistencia.Empleado;
import servicios.interfaces.accesoServicio;
import servicios.interfaces.empleadoServicio;
import utilidades.Ctes;
import utilidades.ManejoExcepciones;
import excepciones.ErrorValidacionException;
```

- Son las librerias utilizadas para el desarrollo del manejo de la funcionalidad.

@Stateful

- Nos indica el tipo de beans que se está manejando.

@Scope(ScopeType.CONVERSATION)

- Nos indica que el bean utilizado es de tipo conversación.

```
@Name("AdmClave")
```

- Nos indica el nombre con el cual será identificado en la página.

```
public class AdmClaveBean implements AdmClave, Serializable
{
   @Logger private Log log;
```

- Nos indica que tiene una notación para la implementación de un log.

```
@In FacesMessages facesMessages;
```

- Nos indica que se realiza una inyección de la clase facesMessages y es utilizada para el manejo de mensajes.

```
@EJB empleadoServicio srvEmpleado;
   @EJB accesoServicio srvAcceso;
```

- Nos indica que se van a utilizar beans asociados para la funcionalidad en este caso son dos servicios.

```
private int value;
   private Empleado empleado;
   private Acceso acceso;
   private String cedulaRest;
   private String claveRest;
   private String cedulaIng;
   private String claveIng;
   private String claveIngC;
   private String cedula;
   private String claveAnt;
   private String claveNew;
   private String claveNewC;
   private String claveStandar;
   private List<Acceso> listaAcceso;
```

- Nos indica las diferentes variables utilizadas.

```
@Begin (join=true)
```

- Nos indica que para iniciar la conversacion los beans asociados se iran uniendo en la conversacion y este sera el inicio de ella.

```
public void begin()
   {
      // implement your begin conversation business logic
      log.info("beginning conversation");
      nuevo();
}
```

- Métodos utilizados para la funcionalidad del bean.

```
public void nuevo(){
         System.out.println("Ingreso a nuevo");
         empleado = new Empleado();
         acceso = new Acceso();
         cedulaRest = "";
         claveRest = "";
         cedulaIng = "";
         claveIng = "";
         claveIngC = "";
         cedula = "";
```

```
        claveAnt = "";
        claveNew = "";
        claveNewC = "";
        listaAcceso = new ArrayList<Acceso>();
        try {
                claveStandar = Ctes.md5("UCE");
        System.out.println("Clave Standar :> "+claveStandar);
        }catch (Exception ex){}

    }

  public void guardar(){
        try{
        System.out.println("Ingreso a guardar");
        if(cedulaIng.equalsIgnoreCase("") ||
cedulaIng.equalsIgnoreCase(null))
                throw new ErrorValidacionException("Debe ingresar una
cedula...");
        if(claveIng.equalsIgnoreCase("") || claveIng.equalsIgnoreCase(null))
                throw new ErrorValidacionException("Debe ingresar una
clave...");
        if(claveIngC.equalsIgnoreCase("") ||
claveIngC.equalsIgnoreCase(null))
                throw new ErrorValidacionException("Debe ingresar una
clave...");
        if(!Ctes.md5(claveIng).equalsIgnoreCase(Ctes.md5(claveIngC)))
                throw new ErrorValidacionException("No coinciden las
claves ingresadas...");
        empleado = srvEmpleado.buscarCedula(cedulaIng);
        if(empleado==null)
                throw new ErrorValidacionException("No existe un empleado
con la cedula:"+cedulaIng);
        listaAcceso = srvAcceso.listar();
        for (Acceso a : listaAcceso) {
                System.out.println("Id a ingresar
:"+empleado.getIdEmpleado());
                System.out.println("Id ingresado
:"+a.getEmpleado().getIdEmpleado());

        if(empleado.getIdEmpleado()==a.getEmpleado().getIdEmpleado())
                        throw new ErrorValidacionException("Ya existe
un empleado registrado con la cedula:"+empleado.getEmpCedula());
                }
        System.out.println("Antes de guardar");
        String clave = Ctes.md5(claveIng);
        System.out.println("Clave MD5 creada :> "+clave);
        Acceso a = new Acceso();
        a.setEmpleado(empleado);
        a.setAccClave(clave);
        srvAcceso.crear(a);
```

```
        System.out.println("Finalizo crear");
        facesMessages.add("Ingreso exitoso...");
        }catch (Throwable ex) {
                ex.printStackTrace();
                String mess = ManejoExcepciones.getMessages(ex, "No se
pudo guardar..., Verifique los datos ingresados");
                        facesMessages.add(ManejoExcepciones.severity,
mess);
        }
    }

    public void reset(){
        try{
        System.out.println("Ingreso a resetear clave");
        if(cedulaRest.equalsIgnoreCase("") ||
cedulaRest.equalsIgnoreCase(null))
                throw new ErrorValidacionException("Debe ingresar una
cedula...");
        empleado = srvEmpleado.buscarCedula(cedulaRest);
        if(empleado==null)
                throw new ErrorValidacionException("No existe un empleado
con la cedula:"+cedulaRest);
        acceso = srvAcceso.buscarCedula(empleado.getIdEmpleado());
        if(acceso==null)
                throw new ErrorValidacionException("No existe un registro
con la cedula:"+cedulaRest);

        System.out.println("Antes de guardar");
        System.out.println("Clave MD5 creada :> "+claveStandar);
        Acceso a = acceso;
        a.setAccClave(claveStandar);
        srvAcceso.actualizar(a);
        System.out.println("Finalizo actualizar");
        facesMessages.add("Reseteo Exitoso...");
        }catch (Throwable ex) {
                ex.printStackTrace();
                String mess = ManejoExcepciones.getMessages(ex, "No se
pudo guardar..., Verifique los datos ingresados");
                        facesMessages.add(ManejoExcepciones.severity,
mess);
        }
    }

    public void guardarClave(){
        try{
        System.out.println("Ingreso a actualizar clave");
        String claveanterior = "";

        if(cedula.equalsIgnoreCase("") || cedula.equalsIgnoreCase(null))
```

```
                throw new ErrorValidacionException("Debe ingresar una
cedula...");
        if(claveAnt.equalsIgnoreCase("") ||
claveAnt.equalsIgnoreCase(null))
                throw new ErrorValidacionException("Debe ingresar una
clave anterior...");
        if(claveNew.equalsIgnoreCase("") ||
claveNew.equalsIgnoreCase(null))
                throw new ErrorValidacionException("Debe ingresar una
clave nueva...");
        if(claveNewC.equalsIgnoreCase("") ||
claveNewC.equalsIgnoreCase(null))
                throw new ErrorValidacionException("Debe ingresar una
clave nueva para confirmar...");
        if(!Ctes.md5(claveNew).equalsIgnoreCase(Ctes.md5(claveNewC)))
                throw new ErrorValidacionException("No coinciden las
claves ingresadas...");
        empleado = srvEmpleado.buscarCedula(cedula);
        if(empleado==null)
                throw new ErrorValidacionException("No existe un empleado
con la cedula:"+cedula);
        acceso = srvAcceso.buscarCedula(empleado.getIdEmpleado());
        if(acceso==null)
                throw new ErrorValidacionException("No existe un registro
con la cedula:"+cedula);

        claveanterior = Ctes.md5(claveAnt);

         System.out.println("Clave almacenada :>"+acceso.getAccClave());
         System.out.println("Clave anterior   :>"+claveanterior);
         if(!acceso.getAccClave().equalsIgnoreCase(claveanterior))
                throw new ErrorValidacionException("La clave anterior no
coincide...");

        System.out.println("Antes de guardar");
        System.out.println("Clave MD5 creada :> "+claveNew);
        Acceso a = acceso;
        a.setAccClave(Ctes.md5(claveNew));
        srvAcceso.actualizar(a);
        System.out.println("Finalizo actualizar");
        facesMessages.add("Actualizacion exitosa...");
        }catch (Throwable ex) {
                ex.printStackTrace();
                String mess = ManejoExcepciones.getMessages(ex, "No se
pudo guardar..., Verifique los datos ingresados");
                        facesMessages.add(ManejoExcepciones.severity,
mess);
        }
   }
```

```
    public String increment()
    {
        log.info("incrementing");
        value++;
        return "success";
    }

    // add additional action methods that participate in this conversation
```

- Método implementado para finalizar la conversación.

```
@End
    public String end()
    {
        // implement your end conversation business logic
        log.info("ending conversation");
        return "home";
    }

    public int getValue()
    {
        return value;
    }

    @Remove
    public void destroy() {}
```

- Implementacion de los getter y los setters.

```
        public Empleado getEmpleado() {
                return empleado;
        }

        public void setEmpleado(Empleado empleado) {
                this.empleado = empleado;
        }

        public String getCedulaIng() {
                return cedulaIng;
        }

        public void setCedulaIng(String cedulaIng) {
                this.cedulaIng = cedulaIng;
        }

        public String getClaveIng() {
                return claveIng;
        }

        public void setClaveIng(String claveIng) {
                this.claveIng = claveIng;
        }
```

```
public String getCedula() {
        return cedula;
}

public void setCedula(String cedula) {
        this.cedula = cedula;
}

public String getClaveAnt() {
        return claveAnt;
}

public void setClaveAnt(String claveAnt) {
        this.claveAnt = claveAnt;
}

public String getClaveNew() {
        return claveNew;
}

public void setClaveNew(String claveNew) {
        this.claveNew = claveNew;
}

public String getClaveNewC() {
        return claveNewC;
}

public void setClaveNewC(String claveNewC) {
        this.claveNewC = claveNewC;
}

public String getClaveIngC() {
        return claveIngC;
}

public void setClaveIngC(String claveIngC) {
        this.claveIngC = claveIngC;
}

public String getCedulaRest() {
        return cedulaRest;
}

public void setCedulaRest(String cedulaRest) {
        this.cedulaRest = cedulaRest;
}

public String getClaveRest() {
        return claveRest;
```

```
    }

    public void setClaveRest(String claveRest) {
        this.claveRest = claveRest;
    }

    public String getClaveStandar() {
        return claveStandar;
    }

    public void setClaveStandar(String claveStandar) {
        this.claveStandar = claveStandar;
    }

    public Acceso getAcceso() {
        return acceso;
    }

    public void setAcceso(Acceso acceso) {
        this.acceso = acceso;
    }

    public List<Acceso> getListaAcceso() {
        return listaAcceso;
    }

    public void setListaAcceso(List<Acceso> listaAcceso) {
        this.listaAcceso = listaAcceso;
    }

}
```

AdmClave.java: Interface ligada al bean.

- Nos indica dentro de que paquete está ubicado

```
package empleado;
```

- Nos indica que librerías se están utilizando.

```
import java.util.List;
import javax.ejb.Local;
import persistencia.Acceso;
import persistencia.Empleado;
```

- Nos indica que es un bean de tipo Local, por lo que podemos definir que es la interface que une el vean de tipo Stateful con la página asociada.

```
@Local
public interface AdmClave
{
```

- Nos indican la definición de los métodos y variables en la interface para que puedan ser utilizados en la página.

```
    public void begin();
```

```
        public String increment();
        public String end();
        public int getValue();
        public void destroy();

    // add additional interface methods here

        public void nuevo();
        public void guardar();
        public void reset();
        public void guardarClave();
        public Empleado getEmpleado();
        public void setEmpleado(Empleado empleado);
        public String getCedulaIng();
        public void setCedulaIng(String cedulaIng);
        public String getClaveIng();
        public void setClaveIng(String claveIng);
        public String getCedula();
        public void setCedula(String cedula);
        public String getClaveAnt();
        public void setClaveAnt(String claveAnt);
        public String getClaveNew();
        public void setClaveNew(String claveNew);
        public String getClaveNewC();
        public void setClaveNewC(String claveNewC);
        public String getClaveIngC();
        public void setClaveIngC(String claveIngC);
        public String getCedulaRest();
        public void setCedulaRest(String cedulaRest);
        public String getClaveRest();
        public void setClaveRest(String claveRest);
        public String getClaveStandar();
        public void setClaveStandar(String claveStandar);
        public Acceso getAcceso();
        public void setAcceso(Acceso acceso);
        public List<Acceso> getListaAcceso();
        public void setListaAcceso(List<Acceso> listaAcceso);
}
```

Webcontent

Dentro de esta carpeta encontramos la interfaz gráfica de todos los vean y con la cual el usuario podrá interactuar con el sistema a continuación se detallara un ejemplo.

admClave.xhtml: Es la página ligada al bean.

- Nos indica la versiondel xhtml utilizado.

```
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
```

```
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:composition xmlns=http://www.w3.org/1999/xhtml
```

- Nos indica las diferentes librerias utilizadas.

```
xmlns:s="http://jboss.com/products/seam/taglib"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:rich="http://richfaces.org/rich"
   xmlns:a4j="http://richfaces.org/a4j"
template="/layout/template.xhtml">
```

- Nos indica la definicion de la estructura que va a ser utilizada.

```
<ui:define name="body">
```

- Hace referencia a una plantilla para el diseño.

```
        <ui:include src="/layout/waitDialog.xhtml">
                <ui:param name="procMessage" value="Procesando..." />
        </ui:include>
```

- Es una propiedad ajax que nos ayuda con iteraciones.

```
        <a4j:queue/>
        <h:form id="admClave">
<a4j:region>
<rich:simpleTogglePanel label="INGRESO CLAVE" id="ingClave">
<s:decorate template="/layout/edit.xhtml" id="idcedulaing">
        <ui:define name="label">CEDULA: </ui:define>
        <h:inputText value="#{AdmClave.cedulaIng}"   maxlength="14"
        readonly="false">
        </h:inputText>
</s:decorate>
<s:decorate template="/layout/edit.xhtml" id="idclaveing">
                <ui:define name="label">CLAVE: </ui:define>
                <h:inputSecret value="#{AdmClave.claveIng}"
                maxlength="20" readonly="false">
                </h:inputSecret>
        </s:decorate>
        <s:decorate template="/layout/edit.xhtml" id="idclaveingc">
                <ui:define name="label">CONFIRMAR CLAVE: </ui:define>
                <h:inputSecret value="#{AdmClave.claveIngC}"
                maxlength="20" readonly="false">
                </h:inputSecret>
        </s:decorate>
</rich:simpleTogglePanel>
<h:panelGrid columns="3">
        <h:commandButton id="guardarc" value="GUARDAR"
        action="#{AdmClave.guardar()}" />
        <h:commandButton id="limpiar" value="LIMPIAR"
        action="#{AdmClave.nuevo()}" />
         <h:commandButton id="salir" value="SALIR"
        action="#{AdmClave.end()}" />
</h:panelGrid>
<rich:simpleTogglePanel label="CAMBIO CLAVE" id="cambioclave">
<s:decorate template="/layout/edit.xhtml" id="idced">
```

```
                <ui:define name="label">CEDULA: </ui:define>
                <h:inputText value="#{AdmClave.cedula}"
                maxlength="14" readonly="false">
                </h:inputText>
                </s:decorate>
        <s:decorate template="/layout/edit.xhtml" id="idclaveant">
                <ui:define name="label">CLAVE ANTERIOR: </ui:define>
                <h:inputSecret value="#{AdmClave.claveAnt}"
                maxlength="20" readonly="false">
                </h:inputSecret>
        </s:decorate>
        <s:decorate template="/layout/edit.xhtml" id="idclavenew">
                <ui:define name="label">CLAVE NUEVA: </ui:define>
                <h:inputSecret value="#{AdmClave.claveNew}"
                maxlength="20" readonly="false">
                </h:inputSecret>
        </s:decorate>
        <s:decorate template="/layout/edit.xhtml" id="idclavenewc">
                <ui:define name="label">CONFIRMAR CLAVE: </ui:define>
                <h:inputSecret value="#{AdmClave.claveNewC}"
                maxlength="20" readonly="false">
                </h:inputSecret>
        </s:decorate>
</rich:simpleTogglePanel>
<h:panelGrid columns="3">
<h:commandButton id="begin" value="GUARDAR"
action="#{AdmClave.guardarClave()}" />
<h:commandButton id="inc" value="LIMPIAR"
action="#{AdmClave.nuevo()}" />
<h:commandButton id="end" value="SALIR" action="#{AdmClave.end()}"
/></h:panelGrid>
rich:simpleTogglePanel label="RESETEAR CLAVE" id="resetclave">
<s:decorate template="/layout/edit.xhtml" id="idcedres">
        <ui:define name="label">CEDULA: </ui:define>
        <h:inputText value="#{AdmClave.cedulaRest}"
        maxlength="14" readonly="false">
        </h:inputText>
</s:decorate>
        </rich:simpleTogglePanel>
        <h:panelGrid columns="3">
         <h:commandButton id="reset" value="RESETEAR"
        action="#{AdmClave.reset()}" />
          <h:commandButton id="rlim" value="LIMPIAR"
        action="#{AdmClave.nuevo()}"  />
          <h:commandButton id="rend" value="SALIR"
        action="#{AdmClave.end()}" />
</h:panelGrid>
</a4j:region>
</h:form>
</ui:define>
```

</ui:composition>

Servicios utilizados:

Dentro del sistema también podemos encontrar servicios desarrollados los cuales permitirán manejo de la funcionalidad del sistema así como las diferentes operaciones con las persistencias, a continuación se detallará un ejemplo se cada uno de ellos.

Servicio utilizados para la funcionalidad.

Como ejemplo se explicará el utilizado para la carga masiva de archivos.

- **ServicioMaterialesAdminBean.java:**

  - Nos indica que se encuentra dentro del paquete srv.

```
package srv;
```

  - Nos indica que librerías se están utilizando.

```
import java.io.File;

import java.io.IOException;

import java.text.ParseException;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import javax.ejb.EJB;

import javax.ejb.Stateless;

import jxl.Sheet;

import jxl.Workbook;

import jxl.read.biff.BiffException;

import persistencia.CabMaterial;

import persistencia.Material;

import servicios.interfaces.cabMaterialServicio;

import servicios.interfaces.materialServicio;
```

  - Esta notación nos dice se está avisando al compilador que suprima advertencias al momento de la ejecución.

@SuppressWarnings("unchecked")

- Nos indica que es un Bean de tipo Stateless.

@Stateless

public class ServicioMaterialesAdminBean implements ServicioMaterialesAdmin {

- Nos indica que se hace una inyección de un bean para ser utilizado en la lógica de funcionalidad.

@EJB cabMaterialServicio srvCabMat;

@EJB materialServicio srvMaterial;

//Cargar archivos bienes desde Excel

- Definición del método.

public List<String> leerArchivoExcel(File archivobien) throws BiffException, IOException, ParseException {

- Implementación utilizada para la lectura de un archivo con formato .xml

List<String> obs = new ArrayList<String>();

Workbook libroExcel = Workbook.getWorkbook(archivobien);

Sheet hoja = libroExcel.getSheet(0);

Integer numeroFilas = hoja.getRows();

for (int f = 1; f < numeroFilas; f++)

Material material = new Material();

CabMaterial cab = new CabMaterial();

cab = srvCabMat.ver(hoja.getCell(0,f).getContents());

if(cab != null)

{

material.setCabMaterial(cab);

material.setMatDescripcion(hoja.getCell(1,f).getContents().toUpperCase());

material.setMatCodigo(hoja.getCell(2,f).getContents().toUpperCase());

material.setMatExistenciasMin(Integer.parseInt(hoja.getCell(3,f).getContents()));

material.setMatFechaIngreso(new Date());

material.setMatCantidadPrestamo(0);

material.setMatCantidadStock(0);

```
material.setMatCantidadTotal(0);

material.setMatSuspendido("ACTIVO");

srvMaterial.crear(material);

obs.add("Se creó el material : "+material.getMatDescripcion()+" con el código :
"+material.getMatCodigo());

}

else

obs.add("Error al crear el material :
"+hoja.getCell(1,f).getContents().toUpperCase()+" con el código :
"+hoja.getCell(2,f).getContents().toUpperCase());

}

libroExcel.close();

return obs;

}

}
```

A este vean se encuentra asociado su interfaz la cual se describe a continuación:

- **ServicioMaterialesAdmin.java**

 Nos indica dentro de que paquete se encuentra.

```
package srv;
```

 Nos indica que librerías se están utilizando.

```
import java.io.File;

import java.io.IOException;

import java.text.ParseException;

import java.util.List;

import javax.ejb.Remote;

import jxl.read.biff.BiffException;
```

 Nos indica que el bean es de tipo @Remote.

```
@Remote


public interface ServicioMaterialesAdmin {
```

- Nos indican los métodos que podrán ser utilizados por medio de la interfaz.

```
public List<String> leerArchivoExcel(File archivobien) throws BiffException,
IOException, ParseException;

}
```

Servicios utilizados para el manejo de la persistencia:

A continuación se describe un ejemplo de un servicio utilizado para el manejo de la persistencia.

accesoServicioBean.java: Servicio utilizado para el manejo de la persistencia Acceso.

- Nos indica dentro de que paquete se encuentra.

```
package servicios.implementaciones;
```

- Nos indica las librerias utilizadas.

```
import java.util.List;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import persistencia.Acceso;

import persistencia.Empleado;

import servicios.interfaces.accesoServicio;
```

- Nos indica que se avisa al compilador que suprima las advertencias al momento de la ejecución.

```
@SuppressWarnings("unchecked")
```

- Nos indica que el bean es de tipo Stateless.

```
@Stateless

public class accesoServicioBean implements accesoServicio

{
```

- Usar la notación @PersistenceContext sirve para obtener una instancia de “EntityManager” con el cual podemos manejar las clases mapeadas a la tablas

```
@PersistenceContext

private EntityManager em;
```

- Nos indica que se implementa el método para actualizar haciendo uso de una instacia del EntityManager.

```
public void actualizar(Acceso entity) {
```

```
em.merge(entity);

}
```

- Nos indica que se implementa el método para buscar haciendo uso de una instancia del EntityManager.

```
public Acceso buscar(Object id) {

return em.find(Acceso.class, id);

}
```

- Nos indica que se implementa el método para crear haciendo uso de una instancia del EntityManager.

```
public void crear(Acceso entity){

em.persist(entity);

}
```

- Nos indica que se implementa el método para eliminar haciendo uso de una instancia del EntityManager.

```
public void eliminar(Object id) {

em.remove(em.getReference(Acceso.class, id));

}

public List<Acceso> listar() {

return em.createQuery("from " + Acceso.class.getName() + " o")

.getResultList();

}

public Acceso buscarCedula(int idempleado) {

String sql = "select o from Acceso o where
o.empleado.idEmpleado=:idempleado";

list<Acceso> list = em.createQuery(sql).setParameter("idempleado",
idempleado).getResultList();

if (list.size() > 0) {

return list.get(0);

} else

return null;

}}
```

Clases utilizadas para la funcionalidad

Dentro del proyecto existen clases creadas con el fin de hacer métodos genéricos y reutilizables como por ejemplo se detalla a continuación la clase AdmEmpresa .

AdmEmpresa.java: Clase utilizada para recuperar el usuario logeado con la finalidad de realizar la auditoria.

- Nos indica dentro de que paquete se encuentra la clase.

```
package utilidades;
```

- Nos indica las librerias utilizadas

```
import java.io.Serializable;

import java.util.Map;

import org.jboss.seam.ScopeType;

import org.jboss.seam.annotations.AutoCreate;

import org.jboss.seam.annotations.Name;

import org.jboss.seam.annotations.Scope;

import org.jboss.seam.annotations.intercept.BypassInterceptors;
```

- Nos indica el nombre con el cual lo podemos referenciar.

```
@Name("admEmpresa")
```

- Nos indica que el besn sera de tipo sesión.

```
@Scope(ScopeType.SESSION)
```

- Nos indica que el bean se creara inmediatamente al ser invocado.

```
@AutoCreate
```

- Nos indica que no podrá ser interceptado en su ejecución.

```
@BypassInterceptors

public class AdmEmpresa  implements Serializable {

private static final long serialVersionUID = 4106563846417726726L;

private String usuario;

public AdmEmpresa() {}

public String getUsuario() {

return usuario;

}
```

```
public void setUsuario(String usuario) {

this.usuario = usuario;

}

}
```

Descripción de la auditoria.

Dentro del proyecto se encuentra definido un paquete llamado auditoria en donde se encuentran las clases utilizadas para la implementación de la auditoria del sistema con la ayuda de la librería envers.

Enterprice Archive (EAR)

EAR (Enterprise Archive) es un formato de archivo utilizado por Java EE para el envasado de uno o más módulos en un solo archivo para que el despliegue de los diferentes módulos en un servidor de aplicaciones que sucede de forma simultánea y coherente. También contieneXML archivos llamados descriptores de despliegue , que se describe cómo implementar los módulos.

Encontramos el archivo de configuración application.xml, el cual nos permite referenciar a los distintos proyectos.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://java.sun.com/xml/ns/javaee"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/application_5.xsd"
	version="5">

	<display-name>Inventory-ear</display-name>

	<!-- JBIDE-4166 Workaround - Make sure we deploy this first, since
this is referenced in xxx-ejb.jar-->
	<module>
		<ejb>jboss-seam.jar</ejb>
	</module>

	<module>
		<web>
			<web-uri>Inventory.war</web-uri>
			<context-root>/Inventory</context-root>
		</web>
	</module>
```

```
    <module>
        <ejb>Inventory-ejb.jar</ejb>
    </module>

</application>
```

Descripción de los paquetes utilizados.

auditoria.- Dentro de este paquete se encontraran todas las clases utilizadas para la implementación de la auditoria y se encuentran configuradas de tal modo que se lo realice por medio de la librería envers.

empleado.- Dentro de este paquete se encuentran las clases que manejan las páginas que contienen información de los usuarios.

excepciones.- Dentro de este paquete se encuentran clases implementadas para el manejo de errores, excepciones, cálculos genéricos, entre otras.

inventario.- Dentro de este paquete se puede encontrar toda la lógica que se implementó para el manejo del inventario desde la autentificación hasta el manejo de menús.

mantenimiento.-Dentro de este paquete se encuentra los beans que manejan a las páginas creadas para la administración de las tablas que no ingresan en la lógica de negocio pero son necesarias para su funcionamiento, más conocidas como lista de valores.

persistencia.- Dentro de este paquete se encuentran las clases correspondientes a la persistencia de la base de datos.

servicios.implementaciones.- Dentro de este paquete se encuentran los servicios utilizados para el manejo de la persistencia, incluyendo los servicios básicos CRUD (creación, lectura, actualización, eliminación).

utilidades.-Dentro de este paquete se encuentran clases genéricas que ayudan a la implementación del sistema.

Diagrama Físico de la Base de Datos.